PLRModels: Un paquete de R para realizar inferencia estadística en modelos de regresión parcialmente lineal

Máster en Técnicas Estadísticas

Alumna:
Ana López Cheda

Director:
Germán Aneiros Pérez

Trabajo Fin de Máster

10 de enero de 2014

El presente documento recoge bajo el título *PLRModels: Un paquete de R para realizar inferencia estadística en modelos de regresión parcialmente lineal*, el trabajo realizado por Dña. Ana López Cheda como Trabajo Fin de Máster del Máster interuniversitario en Técnicas Estadísticas bajo la dirección de D. Germán Aneiros Pérez, Profesor Titular del departamento de Matemáticas de la Universidad de A Coruña. Este proyecto ha sido iniciado con una beca de colaboración en el Departamento de Matemáticas de la Universidad de A Coruña, concedida por el Ministerio de Educación, Cultura y Deporte.

Fdo. : Dña. Ana López Cheda Fdo.: D. Germán Aneiros Pérez

A Coruña, 10 de enero de 2014

Agradecimientos

A mi tutor, Germán Aneiros Pérez, así como a todos los profesores y compañeros del máster.

A Mª Esther López Vizcaíno, funcionaria del IGE, que nos dio a conocer la fuente de los datos utilizada en el paquete *PLRModels* y a Pipo, percebeiro cedeirés que marisquea en Cariño y que nos hizo llegar sus percepciones acerca de la relación entre lo que ocurre con las ventas de percebe en estas localidades.

A mi familia y amigos.

A todos ellos, muchas gracias.

Resumen

En este proyecto se presenta el paquete *PLRModels* de *R*, que implementa herramientas de inferencia estadística aplicadas a los modelos de regresión parcialmente lineal. La primera versión del paquete *PLRModels* fue subida al CRAN el 28 de junio de 2013, y el 1 de enero de 2014 se actualizó a la versión 1.1. Esta librería permite realizar estimación puntual, estimación por intervalos de confianza, selección de la ventana (por validación cruzada y validación cruzada generalizada), contrastes de bondad de ajuste y análisis de la covarianza. Aunque este trabajo está centrado en los modelos de regresión parcialmente lineal, todas las técnicas del paquete están implementadas (siempre que es posible) para modelos lineales, modelos no paramétricos y modelos parcialmente lineales. El software se ilustra a partir de datos simulados y utilizando datos reales sobre ventas y precios del percebe en cuatro localidades gallegas.

Índice

1 Introducción a los modelos de regresión parcialmente lineal

Un modelo de regresión es un modelo estadístico que explica la dependencia de una variable respuesta Y respecto de una o varias variables explicativas X. Se denomina modelo de regresión *simple* cuando considera sólo una variable explicativa; mientras que si presenta dos o más, se conoce como modelo de regresión *múltiple*. La media de la variable respuesta condicionada a la variable explicativa se denomina función de regresión.

Aunque los modelos de regresión fueron utilizados con anterioridad en Astronomía y Física por Laplace y Gauss, el término *regresión* se utilizó por primera vez en un trabajo de Galton en Biología a finales del siglo XIX. Se trataba de un estudio sobre variables antropométricas: al comparar la estatura de padres e hijos, resultó que las estaturas de los hijos cuyos padres tenían una estatura muy superior al valor medio tendían a igualarse a éste, mientras que las de aquellos cuyos padres eran muy bajos tendían a reducir su diferencia respecto a la estatura media; es decir, "regresaban" al promedio. La constatación empírica de esta propiedad se vio reforzada más tarde con la justificación teórica de ese fenómeno.

Dentro de los modelos de regresión, podemos distinguir entre los modelos paramétricos y no paramétricos. En los primeros, la función de regresión depende de una cantidad finita de parámetros. En los segundos, de una cantidad infinita.

El **modelo lineal puro** (caso particular de modelo paramétrico) se define como:

$$Y_i = X_i^T \beta + \varepsilon_i, \qquad i = 1, ..., n \tag{1}$$

donde Y_i es la variable respuesta, $X_i = (x_{i1}, \cdots, x_{ip})^T$ es un vector de variables explicativas, $\beta = (\beta_1, \cdots, \beta_p)^T$ son los parámetros desconocidos que acompañan a las variables y ε_i es el error aleatorio, que suponemos que cumple las hipótesis de homocedasticidad y media nula (clásicamente, también se asume normalidad e independencia).

La estimación del parámetro β se realizará mediante el método de mínimos cuadrados. Se trata de elegir como estimador de β aquel que minimice a la suma de cuadrados:

$$\sum_{i=1}^{n} (Y_i - X_i^T \beta)^2, \tag{2}$$

que en notación matricial podemos expresar:

$$(Y - X\beta)^T (Y - X\beta) = \phi(\beta), \tag{3}$$

donde ϕ es la función objetivo, $Y = (Y_1, \ldots, Y_n)^T$ y $X = (X_1, \ldots, X_n)^T$. Por lo tanto, si derivamos dicha función respecto de β e igualamos a 0, obtenemos:

$$X^T X\beta = X^T Y, \tag{4}$$

conocida como las *ecuaciones normales de regresión* y cuya solución es el estimador de β por mínimos cuadrados:

$$\hat{\beta} = (X^T X)^{-1} X^T Y. \tag{5}$$

Entre las ventajas de este tipo de modelos destaca la facilidad para interpretar el efecto de cada variable explicativa sobre la variable respuesta, pues no hay más que interpretar la estimación del parámetro asociado. Además, una vez construido el modelo de regresión, podemos utilizarlo para realizar predicciones del valor de Y cuando se conoce el valor de X de forma muy sencilla. Sin embargo, es conocido que el principal inconveniente de los modelos de regresión lineal es su falta de flexibilidad, por lo que existen muchas variables cuyas relaciones no se pueden modelizar en términos de estos modelos.

Para evitar el problema de la poca flexibilidad que presentan los modelos de regresión lineales, estudiaremos los **modelos no paramétricos**, definidos como:

$$Y_i = m(T_i) + \varepsilon_i, \qquad i = 1, ..., n \tag{6}$$

siendo T_i un vector de variables explicativas y m una función suave desconocida.

Una aproximación razonable para estimar la curva de regresión m evaluada en t es escoger la media ponderada de las variables respuesta con variable explicativa cercana al punto t. Esto se puede llevar a cabo, por ejemplo, a través del método núcleo. Definimos este procedimiento como:

$$\hat{m}_h(t) = \sum_{i=1}^{n} w_{n,h}(t, T_i) Y_i \tag{7}$$

siendo w una secuencia de pesos que, por ejemplo, para Nadaraya-Watson quedaría de la siguiente forma:

$$w_{n,h}(t, T_i) = \frac{K\left(\frac{t-T_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{t-T_j}{h}\right)} \tag{8}$$

donde h es una ventana de suavización y K es una función núcleo, generalmente simétrica y positiva, que actúa como una función de pesos de forma que otorga mayor importancia a los puntos más cercanos al punto t y menor importancia a los más alejados. El estimador Nadaraya-Watson, esto es, el estimador (7) con pesos (8) es un caso particular del estimador polinómico local de grado p, cuya construcción, en el caso unidimensional, se presenta a continuación.

Para estimar $m(t_0)$ localmente mediante polinomios de grado p consideraremos un problema de mínimos cuadrados ponderados:

$$\min_{\alpha_0,\ldots,\alpha_p} \sum_{i=1}^{n} \left\{ Y_i - \sum_{j=0}^{p} \alpha_j (T_i - t_0)^j \right\}^2 K_h(T_i - t_0), \tag{9}$$

donde p es el grado del ajuste polinomial local. Si denotamos por $(\hat{\alpha}_0, \ldots, \hat{\alpha}_p)$ al minimizador obtenido en (9), se tiene que $\hat{m}(t_0) = \hat{\alpha}_0$.

A partir de la expresión (9), podemos tomar como casos particulares los estimadores tipo núcleo que utilizaremos en este trabajo: para $p = 0$ tendremos el estimador núcleo de Nadaraya-Watson y para $p = 1$ tendremos el estimador lineal local.

De entre las diferentes funciones que se suelen considerar como núcleo, destacan, en el caso unidimensional, las siguientes:

- Quadratic (Epanechnikov) kernel:

$$K(u) = \frac{3}{4}(1 - u^2)\mathbf{1}_{\{|u|\leq 1\}}. \tag{10}$$

- Gaussian kernel:

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}. \tag{11}$$

- Uniform kernel:

$$K(u) = \frac{1}{2}\mathbf{1}_{\{|u|\leq 1\}}. \tag{12}$$

- Triweight kernel:

$$K(u) = \frac{35}{32}(1 - u^2)^3\mathbf{1}_{\{|u|\leq 1\}}. \tag{13}$$

Si bien la elección del núcleo no es muy importante, sí es crucial la elección de un parámetro de suavización apropiado. En las Secciones 5 y

6 trataremos, en el caso de los modelos de regresión parcialmente lineal, el tema de la selección del ancho de banda óptimo, que estudiaremos aplicando dos métodos: *validación cruzada* y *validación cruzada generalizada*.

La flexibilidad de los modelos no paramétricos puros es de mucha ayuda especialmente en un análisis estadístico exploratorio preliminar, cuando en principio no se conoce ninguna información sobre la curva de regresión que sigue el conjunto de datos. Sin embargo, estos modelos presentan algunos inconvenientes.

Por una parte, al contrario de lo que sucedía en los modelos lineales, para este caso puede no ser tan sencillo interpretar el efecto de cada covariable. Por otra parte, los modelos no paramétricos presentan el problema de la maldición de la dimensionalidad: la complejidad del cálculo y de lograr una adecuada representación de los resultados aumenta considerablemente según agregamos variables regresoras; además, las propiedades estadísticas de los estimadores se deterioran rápidamente debido a que el volumen de datos requeridos para mantener un grado de precisión tolerable crece más rápido que la cantidad de variables que se incluyen en el modelo. Es decir, la variabilidad de los estimadores aumentará exponencialmente con la cantidad de variables explicativas. Todo esto da lugar a que, si la cantidad de variables es grande, necesitaríamos cantidades enormes de datos para obtener estimaciones fiables, lo que suele imposibilitar su utilización práctica si se manejan 3 o más covariables.

Para solucionar este problema consideraremos los **modelos de regresión parcialmente lineales**. Se trata de modelos que, además de combinar las ventajas de los modelos paramétrico (lineal) y no paramétrico, atenúan sus inconvenientes. Definimos un modelo de regresión parcialmente lineal como:

$$Y_i = X_i^T \beta + m(T_i) + \varepsilon_i, \qquad i = 1, ..., n \tag{14}$$

siendo Y_i la variable respuesta; $X_i = (x_{i1}, \cdots, x_{ip})^T$ y T_i las variables explicativas; $\beta = (\beta_1, \cdots, \beta_p)^T$ un vector de parámetros desconocidos; m una función desconocida y $\varepsilon_1, \cdots, \varepsilon_n$ los errores aleatorios.

Este modelo se caracteriza por su flexibilidad y por la sencilla interpretación del efecto de cada variable lineal. Además, elude o atenúa el problema de la maldición de la dimensionalidad (en general, la dimensión de T_i es baja), lo que lo convierte en un modelo muy apropiado para diferentes situaciones. Como podemos observar en la expresión (14), un modelo de regresión parcialmente lineal contiene una componente paramétrica y otra

componente no paramétrica.

Aunque existen distintos métodos para estimar β y m, análogamente a lo estudiado para el modelo lineal y el modelo no paramétrico, trabajaremos con la estimación por mínimos cuadrados combinada con la estimación tipo núcleo. Estudiaremos el estimador para β en la Sección 3 y el estimador para m en la Sección 4, así como sus propiedades asintóticas.

Por otra parte, la elección de los parámetros de suavización b y h es uno de los aspectos más importantes para la estimación tipo núcleo. Debemos ser muy cuidadosos al elegir los anchos de banda: con un parámetro de suavizado muy pequeño, el estimador tiende a infrasuavizar; mientras que si la ventana es muy grande, el estimador tiende a sobresuavizar. Trataremos de elegir la ventana óptima por medio de la minimización de algún criterio de error que estudiaremos en las Secciones 5 y 6.

Resumiendo, los modelos parcialmente lineales son muy flexibles, ya que combinan una componente lineal con una componente no paramétrica y se utilizan cuando se cree que la variable respuesta depende linealmente de ciertas variables y se desconoce el tipo de relación con otras variables. Permiten interpretar fácilmente el efecto de cada variable lineal y son preferidos ante un modelo no paramétrico puro debido a que el problema de la maldición de la dimensionalidad se atenúa.

Los modelos de regresión parcialmente lineales fueron ampliamente estudiados en la literatura y aplicados a muy diferentes situaciones con datos reales. Propuestos por primera vez por Engle et al. (1986) en [6], se trataba de estudiar el efecto de las condiciones meteorológicas en la demanda de la electricidad. Usando datos basados en las ventas de electricidad mensuales para cuatro ciudades (variable respuesta Y), los autores consideraban como covariables el precio de la electricidad (X_1), los ingresos (X_2), y la media de temperatura diaria, (T).

Speckman (1988) en [8] estudia otra interesante aplicación a datos reales, donde hace un experimento para determinar si un enjuague bucal de una marca de analgésicos es efectiva para tratar una determinada enfermedad en las encías. En el estudio se considera el índice de SBI, que es una medida indicadora del desgaste de las encías. El experimento consiste en que un grupo de control ($X = 0$) usa sólo agua para enjuagar la boca, mientras que un grupo de tratamiento ($X = 1$) utiliza el analgésico. La variable T es el SBI en el momento inicial y la variable Y es el SBI en la semana 3.

Por otra parte, Schmalensee y Stocker (1999) en [14] también trabajan con un modelo de regresión parcialmente lineal para analizar el consumo doméstico de gasolina en Estados Unidos.

El lector interesado en los modelos de regresión parcialmente lineal puede consultar la monografía de Härdle et al (2000).

2 Paquete PLRModels

El paquete *PLRModels* contiene herramientas de inferencia estadística aplicadas a los modelos de regresión parcialmente lineal. Específicamente, considera funciones para realizar estimación puntual, estimación por intervalos de confianza, selección del ancho de banda (por validación cruzada y validación cruzada generalizada), contrastes de bondad de ajuste y análisis de la covarianza. En todos los procedimientos se considera que la dimensión de la covariable T es 1, y el diseño correspondiente a T es, en algunos casos, fijo y equiespaciado.

Para la estimación tanto de la parte paramétrica como de la parte no paramétrica combinaremos el procedimiento de mínimos cuadrados con métodos de tipo núcleo. En cuanto a los errores del modelo, se permite que sean o bien independientes, o bien una serie temporal.

Cabe destacar que, aunque nuestro interés en este trabajo es centrarnos en los modelos parcialmente lineales, todas las técnicas están implementadas (siempre que es posible) en los 3 tipos de modelos estudiados en la Sección 1: modelos lineales, modelos no paramétricos y modelos parcialmente lineales. De esta forma, aunque en este estudio sólo presentamos las rutinas *plrm* (referidas a los modelos parcialmente lineales), en el paquete están disponibles sus análogas *par* (para los modelos paramétricos) y *np* (para los modelos no paramétricos).

En el paquete *PLRModels* también incluimos 2 conjuntos de datos: *barnacles1* y *barnacles2*, que contienen información mensual sobre la venta de percebes en 2 localidades de Galicia desde el año 2004 hasta el 2013. Toda la información fue descargada de la fuente de datos:
http://www.pescadegalicia.com/ y en nuestro paquete ha sido transformada a través de la función logarítmica.

- ***barnacles1*** es una matriz que guarda en su primera columna el número de ventas (en log(kg)) de percebes en la lonja de Cedeira, en la segunda

columna el precio (en log(euros)/kg) de los percebes en la misma lonja y en la tercera columna almacena el número de ventas (en log(kg)) de percebes en la lonja de Cariño.

- ***barnacles2***, análogamente, es otra matriz que contiene en la primera columna el número de ventas (en log(kg)) de percebes en la lonja de Cangas, en la segunda columna el precio (en log(euros)/kg) de los percebes en la misma lonja y en la tercera columna guarda el número de ventas (en log(kg)) de percebes en la lonja de Baiona.

En las siguientes secciones indicamos el objetivo de cada rutina y explicaremos la metodología utilizada, mostrando además la base teórica que la sustenta. Además, ilustraremos su funcionamiento a través de su aplicación a datos simulados y a datos reales.

Es importante indicar que, debido a la presencia de componente estacional anual (período 12), los datos han tenido que ser diferenciados estacionalmente. Específicamente, las variables incluidas en el modelo:

$$Y_i = X_{i1}\beta_{i1} + X_{i2}\beta_{i2} + m(T_i) + \varepsilon_i, \tag{15}$$

serán

$$Y_i = log(Ventas.Cedeira_i) - log(Ventas.Cedeira_{i-12}), \tag{16}$$

$$X_{i1} = log(Precio.Cedeira_i) - log(Precio.Cedeira_{i-12}), \tag{17}$$

$$X_{i2} = log(Ventas.Cariño_i) - log(Ventas.Cariño_{i-12}) \tag{18}$$

y

$$T_i = i. \tag{19}$$

Nótese que:

$$Y_i \approx \frac{Ventas.Cedeira_i - Ventas.Cedeira_{i-12}}{Ventas.Cedeira_{i-12}}. \tag{20}$$

Algo similar sería aplicable a las variables X_{i1} y X_{i2}. La importancia de estas aproximaciones se verá en la interpretación de los resultados.

3 Programa plrm.beta

3.1 Objetivo

A partir de una muestra $(Y_i, X_{i1}, \cdots, X_{ip}, T_i)$ con $i = 1, \cdots, n$, este programa obtiene estimaciones para β en el modelo de regresión parcialmente lineal mostrado en la expresión (14).

3.2 Metodología

Robinson (1988) en [7] y Speckman (1988) en [8] proponen estimar β de la siguiente forma:

$$\hat{\beta}_b = (\tilde{X}_b^T \tilde{X}_b)^{-1} \tilde{X}_b^T \tilde{Y}_b \tag{21}$$

siendo $\tilde{X}_b = (I - W_b)\mathrm{X}$ e $\tilde{Y}_b = (I - W_b)Y$, donde $W_b = (w_{n,b}(T_i, T_j))_{i,j}$ es una matriz de suavización (n x n) y $b > 0$ el parámetro ventana que controla el grado de suavización, tal que $nb \to \infty$ y $b \to 0$ cuando $n \to \infty$.

Bajo ciertas condiciones (incluyendo independencia en los errores), Robison (1988) [7] y Speckman (1988) [8] muestran que $\hat{\beta}_b$ es $\sqrt{n}$-consistente para β y asintóticamente normal. Esta tasa de convergencia obtenida para muestras independientes e idénticamente distribuidas es todavía la misma para el caso de errores que siguen un proceso de memoria a corto plazo. Gao (1995) [10] investiga la normalidad asintótica y las tasas de convergencia de $\hat{\beta}_b$ para un proceso lineal en los errores; y Aneiros-Pérez y Quintela-del-Río (2001) [18] probaron la normalidad asintótica de este estimador bajo errores α-mixing.

Para el caso de procesos con memoria a largo plazo, Beran y Ghosh (1998) [13] y Aneiros-Pérez et al. (2004) [20] obtienen la $\sqrt{n}$-consistencia de $\hat{\beta}_b$ calculando los órdenes de su sesgo y su varianza, junto con la normalidad asintótica del estimador propiamente normalizado.

Bajo ciertas suposiciones, en Aneiros-Pérez et al. (2004) [20], el Teorema 1 dice que:

$$E(\hat{\beta}_b|X) - \beta = O(b^4 + n^{-2}) + O_p((b^4 + n^{-2})^{1/2}((nb)^{-1} S_{\eta_0,[nb]})^{1/2}); \tag{22}$$

$$var(\hat{\beta}_b|X) = n^{-1}A + o_p(n^{-1}) \tag{23}$$

y

$$n^{1/2}(\hat{\beta}_b - \beta) \xrightarrow{d} N(0, A). \tag{24}$$

A denota una matriz que depende de la estructura de autocovarianzas tanto de X como de ε, y $S_{\eta_0,[nb]}$ es un número real que depende de la autocovarianzas de X.

Este teorema generaliza los Teoremas 2 y 4 de Speckman (1988) [8] y el Teorema 1 de Beran y Ghosh (1998) [13] para el caso de una estructura general de autocovarianzas para X y ε.

3.3 Aplicación a datos simulados

Para mostrar el buen funcionamiento del código, generamos datos simulados de la siguiente forma:

Creamos una matriz de 100 filas con datos pertenecientes a un modelo de regresión parcialmente lineal: dicho modelo tiene 2 variables explicativas, X_1 y X_2, (independientes y generadas a partir de una N(0,1)), el parámetro β es igual a $(0.05, 0.01)$, la función $m(t)$ es del tipo $0.25 * t * (1 - t)$ y los errores ε están generados por un AR(1).

A dicha matriz le aplicamos la función *plrm.beta*, pasándole como argumento el ancho de banda óptimo calculado por el método de validación cruzada generalizada (que estudiaremos en la Sección 6) y almacenamos sus resultados, es decir, guardamos los valores estimados para las dos componentes de la parte paramétrica del modelo.

Repetimos este proceso 1000 veces, de modo que al final de esta simulación, tenemos 1000 valores estimados para cada una de las dos componentes de β. Calculamos la media y la desviación típica de cada una y, si nuestro código funciona correctamente, deberíamos obtener unos valores para el par de medias muy próximos a $(0.05, 0.01)$, los simulados para β, y valores bajos para la desviación típica.

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
b.seq <- seq(0.01, 0.25, length=50)

result <- matrix(NA, 1000, 2)

seed <- 1234
set.seed(seed)

for (i in 1:1000) {

cat(i, "/", 1000, ", ")

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order=c(1,0,0),ar=0.7),sd=0.01,n=n)
y <-  sum + f + epsilon
```

```
data <- cbind(y,x,t)

# Estimamos la componente paramétrica del modelo PLR
# (utilizando GCV)
b.h <- plrm.gcv(data, b.seq=b.seq)$bh.opt
ajuste <- plrm.beta(data=data, b=b.h[1])

result[i,1] <- ajuste$BETA[1]
result[i,2] <- ajuste$BETA[2]
}
```

Ahora calcularemos los valores medios que se obtienen para el parámetro β, después de replicar 1000 veces el experimento; también, sus desviaciones típicas:

```
mean(result[,1])
0.05003473
sd(result[,1])
0.0008992524

mean(result[,2])
0.009996583
sd(result[,2])
[1] 0.0008764019
```

Como podemos observar, la primera componente de β se aproxima mucho al valor con el que la habíamos generado, que era de 0.05; por otra parte, la segunda componente de la parte paramétrica también se aproxima muy bien al 0.01. Además, los valores para las desviaciones típicas en ambos casos son muy bajos. Por lo tanto, podemos concluir que el programa *plrm.beta* junto con el procedimiento de validación cruzada generalizada obtienen resultados satisfactorios.

Una vez mostrado el buen funcionamiento del programa, procedemos a aplicarlo a datos reales.

3.4 Aplicación a datos reales

A continuación aplicaremos la rutina *plrm.beta* a los datos *barnacles1*, modelizados según el modelo de regresión parcialmente lineal:

$$Y_i = X_{i1}\beta_{i1} + X_{i2}\beta_{i2} + m(T_i) + \varepsilon_i, \tag{25}$$

donde sus variables han sido definidas al final de la Sección 2.

Comenzamos cargando el fichero *barnacles1*:

```
data(barnacles1)
data <- as.matrix(barnacles1)
```

Después de haber hecho un análisis de los datos, observamos que presentan componente estacional, por lo que decidimos diferenciarlos. Además, incluimos el instante temporal, T, en la cuarta columna, que dará lugar a la variable cuyo efecto es modelizado no paramétricamente.

```
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))
```

Una vez preparada la matriz de datos que queremos utilizar, aplicamos la función *plrm.beta*. Para eso, calculamos el ancho de banda óptimo por validación cruzada generalizada, que veremos en la Sección 6:

```
b.h <- plrm.gcv(data)$bh.opt
ajuste <- plrm.beta(data=data, b=b.h[1])
ajuste$BETA
```

Y obtenemos como resultado:

```
> ajuste$BETA
             [,1]
[1,] -0.1417538
[2,]  0.4121777
```

Figure 1: Cálculo de la componente paramétrica mediante *plrm.beta*

En la Figura 1 podemos observar que la componente paramétrica estimada para nuestro modelo es: $\beta = (-0.1418, 0.4122)$. Así, por ejemplo, manteniendo fijos los valores de las variables X_2 y T, se tiene que si el cambio experimentado en los precios del percebe en Cedeira, expresado en términos relativos, aumenta en 0.1 unidades (esto es, si el valor de la variable X_1 aumenta en 0.1 unidades), el correspondiente cambio relativo sufrido por las ventas de percebe en Cedeira descenderá en 0.014 unidades (esto es, el valor de la variable Y descenderá en 0.014 unidades). Del mismo modo, manteniendo fijos los valores de las variables X_1 y T, se tiene que si el cambio experimentado en las ventas de percebe en Cariño, expresado en términos relativos, aumenta en 0.1 unidades (esto es, si el valor de la variable X_2 aumenta en 0.1 unidades), el correspondiente cambio relativo sufrido por las ventas de percebe en Cedeira aumentará en 0.041 unidades (esto es, el valor de la variable Y aumentará en 0.041 unidades). Véase el final de la Sección 2 para recordar el contenido de cada variable y su aproximación.

4 Programa plrm.est

4.1 Objetivo

A partir de una muestra $(Y_i, X_{i1}, \cdots, X_{ip}, T_i)$ con $i = 1, \cdots, n$, este programa obtiene estimaciones para β y $m(newt_j)$, con $j = 1, \cdots, J$ en el modelo de regresión parcialmente lineal mostrado en la expresión (14).

4.2 Metodología

Asumiendo que $w_{n,h}(\cdot, \cdot)$ es la función de pesos correspondiente a la estimación polinómica local de grado 0 o 1, se tiene que:

$$\hat{m}_h(t, \beta) = \sum_{i=1}^{n} w_{n,h}(t, T_i)(Y_i - X_i^T \beta) \tag{26}$$

para un parámetro β conocido.

Si en lugar de β insertamos en (26) su estimador (21), resulta el estimador de m propuesto en Robinson (1988) [7] y Speckman (1988) [8]:

$$\hat{m}_h(t, \hat{\beta}_b) = \sum_{i=1}^{n} w_{n,h}(t, T_i)(Y_i - X_i^T \hat{\beta}_b). \tag{27}$$

donde b y h son parámetros de suavización tales que $nb \to \infty$, $nh \to \infty$, $b \to 0$ y $h \to 0$ cuando $n \to \infty$.

Mostramos el Teorema 2 de Aneiros-Pérez et al. (2004) en [20], donde tenemos que, bajo ciertas condiciones:

$$E(\hat{m}_h(t, \hat{\beta}_b)|X) - m(t) = bias(\hat{m}_h(t, \beta))(1 + op(1)) = O_p(h^2), \tag{28}$$

$$var(\hat{m}_h(t, \hat{\beta}_b)|X) = var(\hat{m}_h(t, \beta))(1 + op(1)) = O_p((nh)^{-1} S_{\varepsilon,[nh]}) \tag{29}$$

y

$$C_{n,h,\alpha_\varepsilon,L}(\hat{m}_h(u_1, \hat{\beta}_b) - m(u_1), \cdots, \hat{m}_h(u_k, \hat{\beta}_b) - m(u_k)) \xrightarrow{d} \sigma_{\alpha_\varepsilon,g,K}(N_1, \cdots, N_k), \tag{30}$$

donde, para cada $k \in \mathbb{N}$ fijo, $0 < u_1 < \cdots < u_k < 1$ son puntos fijos arbitrarios y $N_1, \cdots, N_k$ variables estándar normales independientes.

Al igual que en el Teorema 1 de dicho artículo, las condiciones para las bandas dependen de las funciones de autocovarianzas de los procesos de X y ε. Las expresiones (28) y (29) generalizan el Teorema 3 de Speckam (1988)

en [8] para el caso de una estructura general de autocovarianzas para X y ε. Como en Speckman (1988) en [8], usando los resultados de Csorgo y Mielniczuk (1995) en [9] y Deo (1997) en [11] para normalidad, observamos similar comportamiento asintótico para los estimadores no paramétricos $\hat{m}_h(t, \hat{\beta}_b)$ y $\hat{m}_h(t, \beta)$.

4.3 Aplicación a datos simulados

Construiremos matrices con datos simulados (de la misma forma que en la Sección 3.3) y replicaremos 1000 veces el mismo experimento: obtención del estimador de la parte no paramétrica (m evaluado en T) y obtención de los residuos del modelo ajustado, mediante la función *plrm.est*. Aunque este programa también estima la parte paramétrica del modelo, no mostraremos estos resultados al estar ya reflejados en la Sección 3.3.

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
b.seq <- seq(0.01, 0.25, length=50)

error.L2 <- 0
media.residuos2 <- 0

seed <- 1234
set.seed(seed)

for (i in 1:1000) {

cat(i, "/", 1000, ", ")

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order=c(1,0,0),ar=0.7),sd=0.01,n=n)
y <-  sum + f + epsilon
data <- cbind(y,x,t)

# Estimamos la componente paramétrica del modelo PLR
# (utilizando GCV)
b.h <- plrm.gcv(data, b.seq=b.seq)$bh.opt
ajuste <- plrm.est(data=data, b=b.h[1], h=b.h[2])
```

```
error.L2[i] <- sqrt(mean((ajuste$m.t - f)^2))
media.residuos2[i] <- mean(ajuste$residuals^2)
}
```

Ahora calculamos los valores medios que se obtienen, por una parte, para el error L2 y para su desviación típica; y por otra parte, para los residuos al cuadrado y su desviación típica, después de replicar 1000 veces el experimento.

```
mean(error.L2)
0.01194237
sd(error.L2)
0.001888698

mean(media.residuos2)
2.435688e-05
sd(media.residuos2)
9.071602e-06
```

Como podemos observar, para el error L2 obtenemos un valor muy pequeño, junto con una desviación típica considerablemente baja. Por otra parte, la media de los residuos al cuadrado es prácticamente 0, así como su desviación típica. A la vista de estos resultados podemos concluir que nuestro programa funciona de manera correcta.

4.4 Aplicación a datos reales

Aplicamos ahora la rutina *plrm.est* a los mismos datos y modelo que en la Sección 3.4.

```
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))
```

Utilizamos los anchos de banda óptimos b y h obtenidos por validación cruzada generalizada.

```
b.h <- plrm.gcv(data)$bh.opt
ajuste <- plrm.est(data=data, b=b.h[1], h=b.h[2])
```

Como la estimación para la parte paramétrica de este modelo ya fue calculada e interpretada en la Sección 3.4, ahora nos centraremos en la parte no paramétrica.

Primero, mostraremos un gráfico de la función m estimada respecto del tiempo:

```
plot(data[,4], ajuste$m, type="l", xlab="t", ylab="m(t)", lwd=2)
```

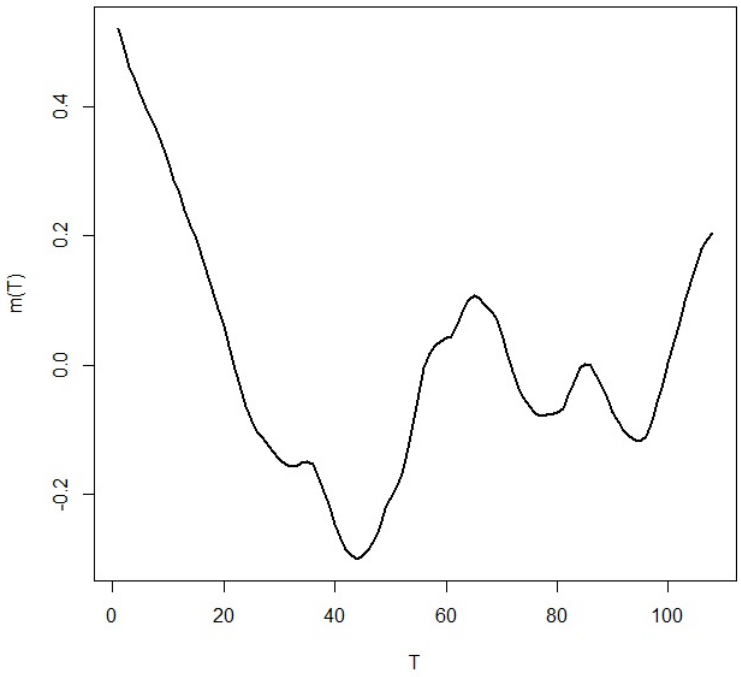

Figure 2: Componente no paramétrica respecto del tiempo calculada mediante *plrm.est*

La Figura 2 muestra la tendencia de los cambios relativos en las ventas de percebes en Cedeira (variable Y) una vez que se les ha extraído el efecto de los cambios relativos en los precios del percebe en Cedeira (variable X_1) y en las ventas de percebes en Cariño (variable X_2). Nótese que si tal efecto no existiese (esto es, si $\beta = (0, 0)$), la Figura 2 estaría indicando distintos comportamientos de la variable Y en función del período en que nos encontremos; por ejemplo, el que la curva sea decreciente y positiva indicaría que las ventas aumentan pero, en términos relativos, cada vez menos; si en cambio la curva es decreciente y negativa lo que ocurriría es que las ventas disminuyen y, en términos relativos, cada vez más;... En el caso esperable de que realmente exista efecto de las variables X_1 y/o X_2 sobre la variable Y (esto es, si $\beta \neq (0, 0)$), la interpretación anterior se mantendría si fijamos los valores de X_1 y X_2 y consideramos como curva la suma de dicho efecto

(que ahora es fijo) con la curva mostrada en la Figura 2.

Continuamos con un gráfico de los valores ajustados frente a los valores reales de la respuesta:

```
plot(data[,1], ajuste$fitted.values, xlab="y", ylab="y.hat",
main="y.hat vs y", lwd=2)
abline(0,1)
```

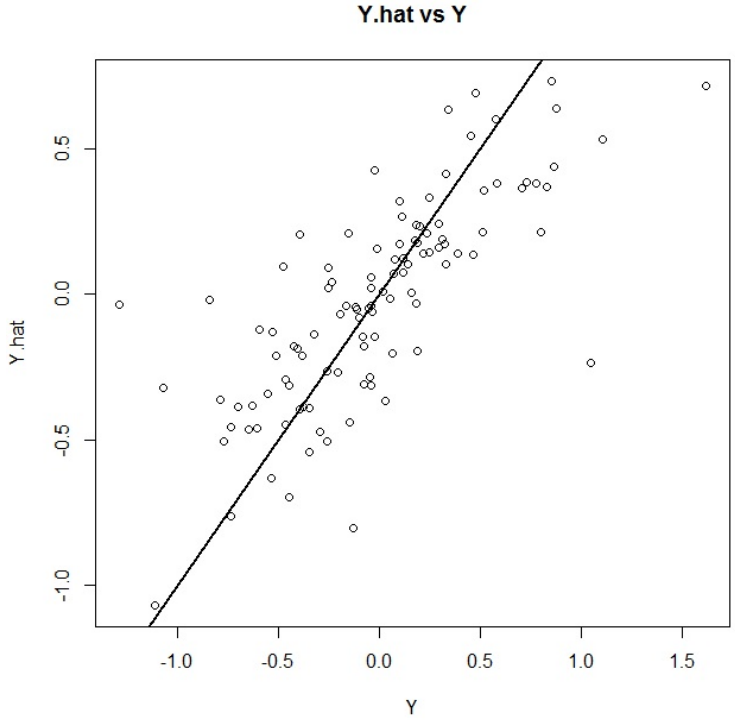

Figure 3: Valores ajustados de Y calculados mediante *plrm.est* vs. valores reales de Y

Como podemos observar en la Figura 3, en general los puntos muestran una tendencia lineal creciente (lo deseable sería que se ajustasen lo mejor posible a la recta mostrada en el gráfico).

Para tener una idea más concreta del grado de ajuste de nuestro modelo, hacemos:

```
mean(ajuste$residuals^2)/var(data[,1])
[1] 0.431517
```

Por lo tanto, tenemos un 43.15% de variabilidad que nuestro modelo no es capaz de explicar, dando lugar a un 56.85% que sí explica.

5 Programa plrm.cv

5.1 Objetivo

Como hemos comentado en la Sección 1, debemos ser muy cuidadosos al elegir las bandas b y h para la expresión (27). Su elección es uno de los aspectos más importantes en la estimación tipo núcleo: si dicho parámetro se toma muy pequeño, tan sólo observaciones muy próximas al punto de estimación intervendrán en el cálculo del estimador, describiendo muy bien comportamientos locales pero obtendremos una curva estimada muy variable. Si por el contrario, dicho parámetro se toma muy grande, las estimaciones en cada punto se verán afectadas por observaciones en puntos alejados de forma que difícilmente se podrán recoger los comportamientos locales, dando lugar a grandes sesgos y como consecuencia obtendremos poca variabilidad. Trataremos de elegir la ventana óptima por medio de la minimización de algún criterio de error; concretamente, a través del método conocido como validación cruzada (modificada).

Por lo tanto, a partir de una muestra $(Y_i, X_{i1}, \cdots, X_{ip}, T_i)$ con $i = 1, \cdots, n$, este programa computa, para cada l_n considerada, un par de bandas óptimo para estimar el modelo de regresión mostrado en (14).

5.2 Metodología

Aneiros-Pérez y Quintela-del-Río (2001) en [17] propusieron seleccionar el parámetro de suavización como el minimizador de la función:

$$CV_{l_n}(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i^T\hat{\beta}_h - \hat{m}_{h,i,l_n}(T_i))^2 \tag{31}$$

siendo $\hat{m}_{h,i,l_n}(\cdot)$ la estimación de $m(\cdot)$ una vez que se han eliminado los $2l_n + 1$ datos más próximos (en el tiempo) a T_i ($l_n = 0, 1, 2, \dots$).

El Teorema 6 de Aneiros-Pérez y Quintela-del-Río (2001) en [17] muestra, bajo ciertas suposiciones, la optimalidad asintótica del procedimiento de validación cruzada:

$$\lim_{n\to\infty} \frac{ASE(\hat{h}_{CV})}{\inf_{h\in H_n} ASE(h)} =_P 1. \tag{32}$$

La rutina implementada generaliza (31) a:

$$CV_{l_n}(b,h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i^T\hat{\beta}_b - \hat{m}_{b,h,i,l_n}(T_i))^2 w(T_i), \tag{33}$$

siendo $w(\cdot)$ una función de pesos introducida para eliminar (o al menos reducir) el problema frontera.

La ventana `b.opt` es usada para estimar β, mientras que el par de ventanas `(b.opt, h.opt)` se utiliza para estimar m. Las estimaciones de β y m se hacen, al igual que en todo este trabajo, combinando el método de mínimos cuadrados ordinarios con la estimación tipo núcleo.

5.3 Aplicación a datos simulados

A partir de una matriz de datos generada de la misma forma que en la Sección 3.3, calcularemos el par de ventanas óptimo para cada réplica. Posteriormente, mostraremos sus medias y sus desviaciones típicas.

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
b.seq <- seq(0.01, 0.25, length=50)

b.opt <- matrix(NA,1000,2)

seed <- 1234
set.seed(seed)

for (i in 1:1000) {

cat(i, "/", 1000, ", ")

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order=c(1,0,0),ar=0.7),sd=0.01,n=n)

y <-  sum + f + epsilon
data <- cbind(y,x,t)
```

```
# Obtenemos el par de bandas óptimo (bandas iguales)
aux <- plrm.cv(data, b.seq=b.seq, step.ln=1, num.ln=2)$bh.opt
b.opt[i,1] <- aux[2,1]
b.opt[i,2] <- aux[2,2]
}
```

Antes de presentar los resultados obtenidos, mostramos los diferentes l_n considerados:

```
aux[1,]
    X1 X2
ln  0  1
```

Por defecto, el programa considera `b=h`, por lo que tomaremos una única ventana óptima que utilizaríamos para estimar β y m.

```
mean(b.opt[,1])
0.01917388
sd(b.opt[,1])
0.005275334

mean(b.opt[,2])
0.04497143
sd(b.opt[,2])
0.03175844
```

En este caso, para $l_n = 0$ tenemos que la media de la ventana óptima es 0.0192 (con desviación típica de 0.0053), mientras que para $l_n = 1$ la media es de 0.0450 (con desviación típica de 0.0318). Un aspecto importante a destacar es que, sobre todo para el primer caso, la desviación típica es considerablemente pequeña respecto del valor de la ventana, por lo que tenemos que todos los anchos de banda resultantes se encuentran muy próximos.

5.4 Aplicación a datos reales

Aplicamos ahora la rutina *plrm.cv* a los mismos datos y modelo que en la Sección 3.4.

```
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))
```

Como comentamos en la Sección 5.3, por defecto el programa considera `b=h`, es decir, obtendremos la misma banda para estimar la parte paramétrica β y la no paramétrica m. Además, eliminaremos $2\ l_n + 1$ observaciones de cada vez.

```
aux <- plrm.cv(data, step.ln=1, num.ln=2)
aux$bh.opt
```

Que nos da como resultado:

```
> aux$bh.opt
          x1       x2
ln 0.000000  1.00000
b  4.454082 23.84184
h  4.454082 23.84184
```

Figure 4: Cálculo del ancho de banda óptimo mediante *plrm.cv*

Como podemos apreciar en la Figura 4, tenemos que el ancho de banda óptimo con $l_n = 0$ es de 4.4541 y para $ln = 1$ es de 23.8418. Nótese que la ventana se va haciendo más grande según aumentamos l_n.

Mostramos ahora una gráfica con los valores de la función CV para cada l_n:

```
par(mfrow=c(2,1))
plot(aux$h.seq,aux$CV[,-2,1], xlab="h", ylab="CV", type="l",
main="ln=0", lwd=2)
plot(aux$h.seq,aux$CV[,-2,2], xlab="h", ylab="CV", type="l",
main="ln=1", lwd=2)
```

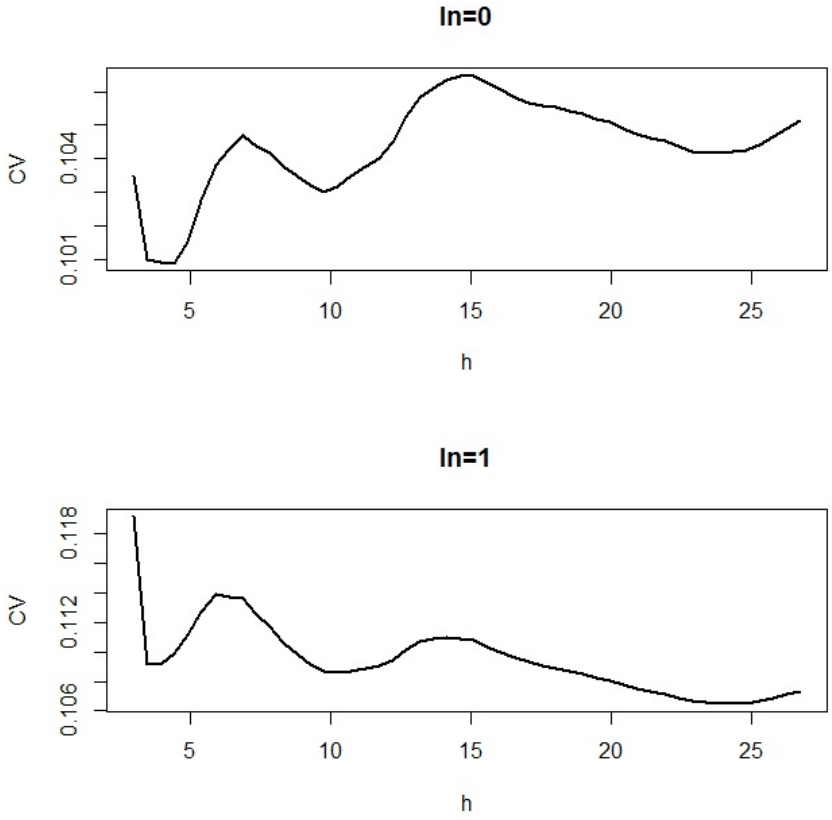

Figure 5: Función CV obtenida mediante *plrm.cv*

A la vista de la Figura 5, podemos corroborar los anchos de banda óptimos que habíamos calculado antes. Observamos que para $l_n = 0$, el mínimo de la función CV se encuentra cuando $h \simeq 4$, mientras que en la gráfica de $l_n = 1$, tenemos el mínimo con $h \simeq 24$.

6 Programa plrm.gcv

6.1 Objetivo

En este apartado mostraremos otro procedimiento para elegir las bandas b y h en la expresión (27), ya que como comentamos, su elección es extremadamente importante cuando trabajamos con la estimación tipo núcleo. En este caso, estudiaremos el procedimiento de validación cruzada generalizada, introducido originalmente por Craven y Wahba (1979) [2] y Golub et al. (1979) [3] para suavización por *splines* y por Rice (1984) [4] para suavización tipo núcleo.

A partir de una muestra $(Y_i, X_{i1}, \cdots, X_{ip}, T_i)$ con $i = 1, \ldots, n$, este programa computa un par de bandas óptimo `(b.opt, h.opt)` para estimar el modelo de regresión mostrado en (14).

La ventana `b.opt` es usada para estimar β, mientras que el par de ventanas `(b.opt, h.opt)` se utiliza para estimar m. Este procedimiento, como todos en este trabajo, realiza las estimaciones de β y m con mínimos cuadrados ordinarios combinados con estimación tipo núcleo.

6.2 Metodología

La rutina implementada generaliza lo mostrado en Speckman (1988) [8], donde se muestra un procedimiento que intenta proporcionar una estimación de b basada en los datos, la cual minimice:

$$R(b) = \frac{1}{n} \parallel E(Y) - \hat{Y}_b \parallel^2 \tag{34}$$

para el modelo (14), donde suponemos que $\hat{Y}_b$ es el estimador de $E(Y)$ (dependiente de b) dado por:

$$\hat{Y}_b = X\hat{\beta}_b + \hat{m}_b = X\hat{\beta}_b + W_b(Y - X\hat{\beta}_b) = W_b Y + \tilde{X}_b\hat{\beta}_b. \tag{35}$$

La definición de validación cruzada generalizada requiere la utilización de la llamada *hat matrix*, que denotaremos por $A(b)$, tal que: $\hat{Y}_b = A(b)Y$. La función de validación cruzada generalizada viene dada por:

$$GCV(b) = \frac{RSS(b)}{[1 - n^{-1}tr(A(b))]^2}, \tag{36}$$

donde $RSS(b)$ denota la suma de cuadrados residual:

$$RSS(b) = n^{-1} \parallel (I - A(b))Y \parallel^2 .$$

El procedimiento de validación cruzada generalizada consiste en seleccionar la banda b que minimiza la expresión (36).

Para nuestro caso, permitiremos 2 parámetros de suavización en vez de uno sólo, según lo estudiado en Aneiros-Perez et al. (2004) [20].

6.3 Aplicación a datos simulados

Generamos una matriz de datos de la misma forma que para los datos simulados en la Sección 3.3 y calculamos el par de ventanas óptimo para cada réplica, mostrando posteriormente sus medias y sus desviaciones típicas.

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
b.seq <- seq(0.01, 0.25, length=50)

b.opt <- 0

seed <- 1234
set.seed(seed)

for (i in 1:1000) {

cat(i, "/", 1000, ", ")

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order=c(1,0,0),ar=0.7),sd=0.01,n=n)
y <-  sum + f + epsilon
data <- cbind(y,x,t)

# Obtenemos el par de bandas óptimo (bandas iguales)
b.opt[i] <- plrm.gcv(data, b.seq=b.seq)$bh.opt[1]
}
```

Ahora mostramos la media y la desviación típica del ancho de banda óptimo después de haber hecho 1000 réplicas del experimento. Al igual que la función de validación cruzada, ésta también toma por defecto `b=h`, por lo que tendremos una única ventana para estimar β y m.

```
mean(b.opt)
0.01988898
sd(b.opt)
0.005084725
```

Tenemos que la media de la ventana óptima es de 0.0199, con desviación típica de 0.0051. Este resultado es muy similar al obtenido para el caso de validación cruzada con $l_n = 0$.

6.4 Aplicación a datos reales

Aplicamos ahora la rutina *plrm.gcv* a los mismos datos y modelo que en la Sección 3.4.

```
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))
```

Como, por defecto `b=h`, obtendremos la misma banda para estimar la parte paramétrica y la no paramétrica del modelo.

```
aux <- plrm.gcv(data)
aux$bh.opt
```

```
> aux$bh.opt
[1] 9.785714 9.785714
```

Figure 6: Cálculo del ancho de banda óptimo mediante *plrm.gcv*

A la vista de la Figura 6, tenemos que la ventana óptima para calcular β y m por validación cruzada generalizada es 9.7857.

Análogamente a lo estudiado en la Sección 5.4, también mostraremos una gráfica de la función GCV para corroborar el ancho de banda óptimo.

```
plot(aux$h.seq, aux$GCV, xlab="h", ylab="GCV", type="l", lwd=2)
```

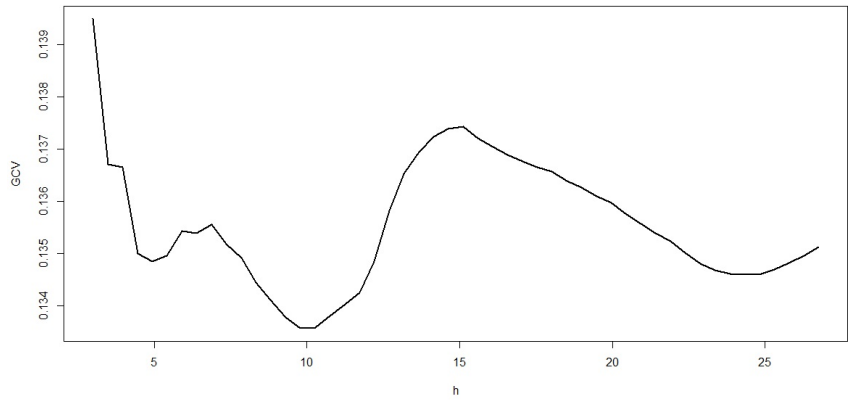

Figure 7: Función GCV obtenida mediante *plrm.gcv*

Como podemos observar en la Figura 7, el mínimo de la función GCV se encuentra alrededor del 10, por lo que ese será el valor de la ventana óptima.

7 Programa plrm.gof

7.1 Objetivo

En cualquier análisis de regresión resulta importante reducir la dimensión del vector de variables explicativas o, en general, simplificar el modelo. Para un modelo de regresión parcialmente lineal, podemos contrastar si la parte paramétrica es una β_0 dada y por otro lado, si la parte no paramétrica m se puede considerar igual a una determinada función m_0.

El programa *plrm.gof*, a partir de una muestra $(Y_i, X_{i1}, \cdots, X_{ip}, T_i)$ con $i = 1, \ldots, n$, en un modelo del tipo (14), realiza los siguientes contrastes de hipótesis:

$$\begin{cases} H_{0\beta} : \beta = \beta_0 \\ H_{1\beta} : \beta \neq \beta_0 \end{cases} \tag{37}$$

y

$$\begin{cases} H_{0m} : m = m_0 \\ H_{1m} : m \neq m_0 \end{cases} \tag{38}$$

7.2 Metodología

Los procedimientos implementados generalizan a las expresiones (9) y (10) de González-Manteiga y Aneiros-Pérez (2003) en [19], permitiendo alguna condición de dependencia en X e incluyendo una función de pesos en el estadístico utilizado en (38). El objetivo de incluir tal función de pesos es eliminar (o al menos, reducir) los problemas con el efecto frontera al estimar $m(T_i)$.

Bajo condiciones generales, $\hat{r}_n(X^T, T) = X^T\hat{\beta}_b + \hat{m}_h(T, \hat{\beta}_b)$ es un estimador consistente para la función de regresión $r(X^T, T) = X^T\beta + m(T)$. Además, bajo las hipótesis nulas $H_{0\beta}$ y H_{0m}, $r(X^T, T)$ es consistentemente estimado mediante $\hat{r}_{n,\beta_0}(X^T, T) = X^T\beta_0 + \hat{m}_h(T, \beta_0)$ y $\hat{r}_{n,m_0}(X^T, T) = X^T\hat{\beta}_b + m_0(T)$, respectivamente. Por lo tanto, tenemos que un estadístico natural para contrastar la parte paramétrica (37) es:

$$d^2_\beta(\hat{r}_n, H_{0\beta}) = n^{-1}\sum_{i=1}^{n}(\hat{r}_n(X_i^T, T_i) - \hat{r}_{n,\beta_0}(X_i^T, T_i))^2 =$$

$$= (\hat{\beta}_b - \beta_0)^T \left(n^{-1}\tilde{X}_h^T\tilde{X}_h\right)(\hat{\beta}_b - \beta_0)$$

y para la parte no paramétrica (38) es:

$$d^2_m(\hat{r}_n, H_{0m}) = n^{-1}\sum_{i=1}^{n}(\hat{r}_n(X_i^T, T_i) - \hat{r}_{n,m_0}(X_i^T, T_i))^2 w(T_i) =$$

$$= n^{-1} \sum_{i=1}^{n} (\hat{m}_h(T_i, \hat{\beta}_b) - m_0(T_i))^2 w(T_i).$$

A continuación mostramos los comportamientos asintóticos de las distancias $d^2_\beta(\hat{r}_n, H_{0\beta})$ y $d^2_m(\hat{r}_n, H_{0m})$. Comenzamos con el de la parte paramétrica:

Bajo ciertas condiciones contempladas en [19], tenemos que bajo la hipótesis nula $H_{0\beta}$:

$$F(b,h) \equiv \frac{n d^2_\beta(\hat{r}_n, H_{0\beta})}{\sigma^2_\varepsilon} = \frac{(\hat{\beta}_b - \beta_0)^T \left(\tilde{X}_h^T \tilde{X}_h \right) (\hat{\beta}_b - \beta_0)}{\sigma^2_\varepsilon} \xrightarrow{d} \chi^2_p, \qquad (39)$$

donde χ^2_p denota la distribución Chi-cuadrado con p grados de libertad; mientras que bajo $H_{1\beta}$, tenemos que $F(b,h) \to \infty$ cuando $n \to \infty$.

Para el estudio del comportamiento asintótico de $d^2_m(\hat{r}_n, H_{0m})$, bajo ciertas condiciones mostradas en [19] y asumiendo además diseño fijo y equiespaciado en $[0,1]$ para T, tenemos que bajo la hipótesis nula H_{0m}:

$$\sqrt{n^2 h} \left(d^2_m(\hat{r}_n, H_{0m}) - \frac{\sum_{s=-\infty}^{\infty} r_\varepsilon(s) \int K^2 \int w}{nh} \right) \xrightarrow{d} N(0, \sigma^2_d), \qquad (40)$$

donde $\sigma^2_d = 2(\sum_{k=-\infty}^{\infty} r_\varepsilon(k))^2 \int (K * K)^2$ y $K * K$ denota la convolución de K consigo misma. $r_\varepsilon(\cdot)$ es la función de autocovarianzas de $\{\varepsilon_i\}$.

Con lo que rechazaremos $H_{0\beta}$ para un nivel de significación α si:

$$d^2_\beta(\hat{r}_n, H_{0\beta}) > n^{-1} \hat{\sigma}^2_\varepsilon \Psi^{-1}(1-\alpha), \qquad (41)$$

donde $\hat{\sigma}^2_\varepsilon = n^{-1} \sum_{i=1}^{n} \hat{\varepsilon}^2_i$, con $\hat{\varepsilon}_i = Y_i - \hat{r}_n(X_i^T, t_i)$, y Ψ es la función de distribución de χ^2_p.

Análogamente rechazaremos H_{0m} para un nivel de significación α si:

$$d^2_m(\hat{r}_n, H_{0m}) > n^{-1} h^{-1/2} \hat{\sigma}_d \Phi(1-\alpha) + n^{-1} h^{-1} \hat{S} \int K^2 \int w, \qquad (42)$$

donde $\hat{S}$ es una estimación de $S = \sum_{k=-\infty}^{\infty} r_\varepsilon(k)$, $\hat{\sigma}^2_d = 2\hat{S}^2 \int (K * K)^2$ y Φ es la distribución Normal estándar. En el programa *plrm.gof*, $\hat{S}$ se construye en base a un modelo ARMA ajustado a los residuos.

7.3 Aplicación a datos simulados

Para mostrar el buen funcionamiento del programa *plrm.gof*, comenzaremos construyendo una matriz de datos de forma análoga a la Sección 3.3. A dicha matriz le aplicaremos dos contrastes de hipótesis: primero $H_{0\beta} : \beta = \beta_0$ y $H_{0m} : m = m_0$, siendo β_0 y m_0 las verdaderas β y m, respectivamente; y posteriormente, $H_{0\beta} : \beta = 0$ y $H_{0m} : m = 0$, es decir, comparando con los valores por defecto.

Crearemos dos matrices (`contraste1` y `contraste2`) que almacenarán los p-valores correspondientes para cada contraste. Repetiremos este proceso 1000 veces y luego calcularemos la proporción de fallos de ambos contrastes:

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
b.seq <- seq(0.01, 0.25, length=50)

f.function <- function(u) {0.25*u*(1-u)}

contraste1 <- matrix(NA, 1000, 2)
colnames(contraste1) <- c("Paramétrico","No paramétrico")

contraste2 <- matrix(NA, 1000, 2)
colnames(contraste2) <- c("Paramétrico","No paramétrico")

seed <- 1234
set.seed(seed)

for (i in 1:1000) {

  cat(i, "/", 1000, ", ")

  x <- matrix(rnorm(200,0,1), nrow=n)
  sum <- x%*%beta
  epsilon <- arima.sim(list(order=c(1,0,0), ar=0.7),sd =0.01,n=n)
  y <-  sum + f + epsilon
  data <- cbind(y,x,t)

  ## Ejemplo 1: H0 cierta
  b.h <- plrm.gcv(data, b.seq=b.seq)$bh.opt
```

```
  result1 <- plrm.gof(data, beta0=c(0.05, 0.01), m0=f.function,
  b.seq=b.h[1], h.seq=b.h[2], time.series=TRUE)
  contraste1[i,1] <- result1$parametric.test$p.v
  contraste1[i,2] <- result1$nonparametric.test$p.v

  ## Ejemplo 2: H0 falsa
  result2 <- plrm.gof(data, b.seq=b.h[1], h.seq=b.h[2],
  time.series=TRUE)
  contraste2[i,1] <- result2$parametric.test$p.v
  contraste2[i,2] <- result2$nonparametric.test$p.v
}
```

Ahora comprobamos los resultados para el primer ejemplo: cuando H_0 es cierta en la parte paramétrica y en la no paramétrica. Es decir, tenemos los contrastes $H_{0\beta} : \beta = \beta_0$ y $H_{0m} : m = m_0$. Calculamos la proporción de veces que obtenemos un p-valor< 0.05; es decir, obtendremos el nivel de significación $P(ERROR\,TIPO\,I)$ empírico del contraste, siendo el nivel $\alpha = 0.05$.

```
alpha <- 0.05

# Contraste paramétrico
sum(contraste1[,1]<alpha)/1000
0.034

# Contraste no paramétrico
sum(contraste1[,2]<alpha)/1000
0.007
```

Nótese que, como era de esperar, el nivel de significación empírico del contraste paramétrico está más próximo al real que el del contraste no paramétrico.

Continuamos con el segundo ejemplo: H_0 es falsa en la parte paramétrica y en la no paramétrica. Concretamente, los contrastes que estamos realizando son: $H_{0\beta} : \beta = 0$ y $H_{0m} : m = 0$. Obtenemos la proporción de veces que el programa da como salida un p-valor> 0.05; es decir, obtendremos la proporción de veces que se comete un error de tipo 2.

```
# Contraste paramétrico
sum(contraste2[,1]>alpha)/1000
0

# Contraste no paramétrico
sum(contraste2[,2]>alpha)/1000
0
```

Para ambos contrastes obtenemos un porcentaje de error igual a 0 (nótese que habíamos hecho el experimento con 1000 réplicas), es decir, el contraste en ningún caso aceptó H_0 siendo H_1 cierta.

Cabe destacar que el valor real de β es muy próximo a 0 (recordamos que en la simulación le habíamos dado valores de (0.05, 0.01)), y aún así el contraste no acepta en ninguno de los casos que la parte paramétrica de modelo sea igual a 0. Algo similar ocurre con el contraste no paramétrico, lo que nos lleva a considerar que el programa *plrm.gof* funciona razonablemente bien.

7.4 Aplicación a datos reales

Aplicamos ahora la rutina *plrm.gof* a los mismos datos y modelo que en la Sección 3.4. Concretamente, contrastaremos las hipótesis nulas `beta0=0` y `m0=0`. Es decir, estaríamos comparando si el modelo tiene una parte paramétrica y por otra parte, si la función de la parte no paramétrica se podría considerar igual a función constante 0.

Preparamos la matriz de datos como en los anteriores ejemplos:

```
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))
```

Y le aplicamos la función *plrm.gof*:

```
plrm.gof(data)
```

```
> plrm.gof(data)
$parametric.test
      b        Q          p.v
1   5.4 47.78322 4.207323e-11
2   7.8 52.30244 4.392042e-12
3  10.2 57.16547 3.860245e-13
4  12.6 64.08210 1.210143e-14
5  15.0 70.45827 5.551115e-16
6  17.4 74.60030 1.110223e-16
7  19.8 76.21042 0.000000e+00
8  22.2 76.74665 0.000000e+00
9  24.6 76.46407 0.000000e+00
10 27.0 76.14834 0.000000e+00

$nonparametric.test
      b    h           Q Q.normalised          p.v
1   5.4  5.4 0.025149188     3.186633 0.0007196969
2   7.8  7.8 0.019378056     3.193239 0.0007034329
3  10.2 10.2 0.015496949     3.006093 0.0013231385
4  12.6 12.6 0.013338274     2.982338 0.0014302805
5  15.0 15.0 0.011853633     2.981433 0.0014345143
6  17.4 17.4 0.010392606     2.839754 0.0022574193
7  19.8 19.8 0.008959468     2.585983 0.0048550871
8  22.2 22.2 0.007623026     2.271273 0.0115652196
9  24.6 24.6 0.006416895     1.931446 0.0267139634
10 27.0 27.0 0.005357079     1.592800 0.0556025121
```

Figure 8: Resultados de *plrm.gof* para datos reales

Estudiando la parte superior de la Figura 8 (correspondiente al test paramétrico), vemos que para todos los anchos de banda, los p-valores obtenidos son muy pequeños, por lo que podemos concluir que el parámetro β no puede ser igual a 0.

Por otro lado, mirando a la parte inferior de la Figura 8 (test no paramétrico), tenemos que todos los p-valores son menores que 0.05 (excepto el último, que se pasa por muy poco). Por tanto, también deducimos que la función m no se puede considerar igual a 0.

Esto da lugar a que claramente para estos datos podemos considerar un modelo de regresión con una parte paramétrica y otra no paramétrica, ya que resulta que ambas, según este contraste, son diferentes de 0.

8 Programa plrm.ci

8.1 Objetivo

Esta rutina obtiene sendos intervalos de confianza para el valor $a^T\beta$ mediante bootstrap y mediante la distribución asintótica, a partir de una muestra $(Y_i, X_{i1}, \cdots, X_{ip}, T_i)$ con $i = 1, \cdots, n$, perteneciente a un modelo del tipo (14), donde:

$$a = (a_1, \cdots, a_p)^T \tag{43}$$

es un vector conocido que elige el usuario y

$$\beta = (\beta_1, \cdots, \beta_p)^T \tag{44}$$

es un parámetro desconocido.

De esta forma, si a es del tipo $(0_1, \cdots, 0_{i-2}, 0_{i-1}, 1_i, 0_{i+1}, \cdots, 0_p)$ (es decir, un vector con $p-1$ ceros y sólo un uno en la posición i), podemos comprobar si la componente i de β es significativa. Por ejemplo, tomando $a = (1_1, 0_2, \cdots, 0_p)$, estudiaremos el intervalo de confianza para la primera componente del vector β.

8.2 Metodología

En este apartado, estudiaremos los dos procedimientos utilizados en la rutina *plrm.ci* para obtener los intervalos de confianza. Comenzamos con el basado en la **distribución asintótica**.

Como vimos en la Sección 3.2, el Teorema 1 de Aneiros-Pérez et al. (2004) en [20] dice que, bajo ciertas condiciones:

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{d} N(0, A), \tag{45}$$

siendo A cierta matriz que depende de la estructura de autocovarianzas de X y ε.

Por tanto, tenemos que:

$$\sqrt{n}a^T(\hat{\beta} - \beta) \xrightarrow{d} N(0, a^T A a). \tag{46}$$

Entonces, un intervalo de confianza para $a^T\beta$ obtenido a través de la distribución asintótica (46) y utilizando una estimación de A, $\hat{A}$, es:

$$(a^T\hat{\beta} - z_{\alpha/2}\hat{D}_T, a^T\hat{\beta} + z_{\alpha/2}\hat{D}_T), \tag{47}$$

donde $\hat{D}_T = \sqrt{\frac{a^T \hat{A} a}{n}} = \sqrt{a^T (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T \hat{V}_\varepsilon \tilde{X} (\tilde{X}^T \tilde{X}^{-1}) a}$, siendo $\hat{V}_\varepsilon$ una estimación de la matriz de varianzas-covarianzas de ε obtenida en base a un modelo ARMA ajustado a los residuos.

Continuamos con el estudio del procedimiento **bootstrap**.

You y Zhou (2005) en [22] proponen un método bootstrap para obtener un intervalo de confianza para $a^T\beta$. Se basan en un modelo parcialmente lineal con errores AR(1), utilizan un estimador generalizado para β y prueban la consistencia del bootstrap. Nosotros imitaremos dicho método, pero utilizando el estimador ordinario para β y asumiendo errores ARMA.

Por lo tanto, aplicaremos el bootstrap en 7 pasos que describimos a continuación:

1. Dada la muestra inicial (Y_i, X_i^T, T_i) con $i = 1, \cdots, n$, obtenemos $\hat{\beta}$ y $\hat{m}$ a partir de las expresiones (21) y (27), respectivamente.
2. Buscamos el modelo ARMA(p,q) que mejor se ajuste a los residuos estimados $\hat{\varepsilon}_i = Y_i - X_i^T\hat{\beta} - \hat{m}(T_i)$, con $i = 1, \cdots, n$. A continuación, obtenemos los residuos del ARMA, $\hat{e}_i$, y los estandarizamos, dando lugar a $e_i = \hat{e}_i - \bar{\hat{e}}$. Además, hacemos un análisis de residuos para comprobar la idoneidad de ARMA ($\hat{e}_i$ incorrelados y con media 0).
3. Remuestreamos el ruido blanco e_i: una vez eliminados los p primeros elementos del vector, obtenemos la muestra de innovaciones bootstrap e_i^*, para $-N \leq i \leq n$ (para un N suficientemente grande).
4. Obtenemos la muestra de residuos con réplicas bootstrap: $\hat{\varepsilon}_i^* = \sum_{j=0}^{N} \hat{\rho}_j e_{i-j}^*$ con $i = 1, \cdots n$, donde $\hat{\rho}_j$ representa los coeficientes de la expresión causal del ARMA.
5. Con los residuos anteriores, calculamos $Y_i^* = X_i^T\hat{\beta} + \hat{m}(T_i) + \hat{\varepsilon}_i^*$, con $i = 1, \cdots, n$.
6. A partir de los datos (Y_i^*, X_i^T, T_i) obtenemos $\hat{\beta}^*$.
7. Para un valor grande de B, repetimos del paso 3 al paso 6 B veces para obtener réplicas bootstrap del estimador $\hat{\beta}^*$.

El intervalo de confianza al $100(1-\alpha)\%$ que obtendremos para el valor $a^T\beta$ aplicando la función *plrm.ci* será el mostrado en la expresión (48):

$$\left(a^T\hat{\beta} - \frac{1}{\sqrt{n}} z_{(1-\alpha/2)}, a^T\hat{\beta} - \frac{1}{\sqrt{n}} z_{(\alpha/2)} \right), \tag{48}$$

siendo z la distribución de $\sqrt{n}a^T(\hat{\beta}^* - \hat{\beta})$.

8.3 Aplicación a datos simulados

Para mostrar el correcto funcionamiento de la rutina *plrm.ci*, generamos datos simulados de la misma forma que en la Sección 3.3 y calcularemos los intervalos de confianza para cada una de las dos componentes de β.

Construiremos cuatro matrices (`intervalo1_boots`, `intervalo1_AD`, `intervalo2_boots` e `intervalo2_AD`) que almacenarán los valores inferiores y superiores de cada intervalo de confianza obtenido por cada método. Debido al gran coste computacional que ese programa requiere, repetiremos este proceso 100 veces y no 1000 (como teníamos para los otros ejemplos con datos simulados). Además, y con el fin de agilizar los cálculos, sólo tomaremos una ventana por validación cruzada generalizada al principio de la simulación y trabajaremos con ella en todas las réplicas.

Por último, analizaremos medidas resumen de los intervalos de confianza obtenidos.

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
b.seq <- seq(0.01, 0.25, length=50)

intervalo1_boots <- matrix(NA, 100, 2)
colnames(intervalo1_boots) <- c("Ext_Inferior","Ext_Superior")
intervalo1_AD <- matrix(NA, 100, 2)
colnames(intervalo1_AD) <- c("Ext_Inferior","Ext_Superior")

intervalo2_boots <- matrix(NA, 100, 2)
colnames(intervalo2_boots) <- c("Ext_Inferior","Ext_Superior")
intervalo2_AD <- matrix(NA, 100, 2)
colnames(intervalo2_AD) <- c("Ext_Inferior","Ext_Superior")

seed <- 1234
set.seed(seed)

for (i in 1:100) {
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order=c(1,0,0),ar=0.7),sd = 0.01,n=n)
y <-  sum + f + epsilon
data <- cbind(y,x,t)
```

```
if (i==1) {b.h <- plrm.gcv(data, b.seq=b.seq)$bh.opt; b1 <- b.h[1]}

result1 <- plrm.ci(data, b1=b1, b2=b1, a=c(1,0), CI="all")
intervalo1_boots[i, 1] <- result1$Bootstrap$ci_inf
intervalo1_boots[i, 2] <- result1$Bootstrap$ci_sup
intervalo1_AD[i, 1] <- result1$AD$ci_inf
intervalo1_AD[i, 2] <- result1$AD$ci_sup

result2 <- plrm.ci(data, b1=b1, b2=b1, a=c(0,1), CI="all")
intervalo2_boots[i, 1] <- result2$Bootstrap$ci_inf
intervalo2_boots[i, 2] <- result2$Bootstrap$ci_sup
intervalo2_AD[i, 1] <- result2$AD$ci_inf
intervalo2_AD[i, 2] <- result2$AD$ci_sup
}
```

Analizamos los resultados obtenidos para el primero de los ejemplos: cálculo del intervalo de confianza para β_1. Recordemos que dicha componente tiene un valor real igual a 0.05. Comenzamos obteniendo la media y desviación típica de cada uno de los extremos de los intervalos:

```
# Primera componente de beta mediante bootstrap
# Extremo inferior
mean(intervalo1_boots[,1])
0.04838039
sd(intervalo1_boots[,1])
0.0002351041

# Extremo superior
mean(intervalo1_boots[,2])
0.05277457
sd(intervalo1_boots[,2])
0.0001183661


# Primera componente de beta mediante distribución asintótica
# Extremo inferior
mean(intervalo1_AD[,1])
0.0483203
sd(intervalo1_AD[,1])
0.0002039233

# Extremo superior
mean(intervalo1_AD[,2])
0.0529894
```

```
sd(intervalo1_AD[,2])
6.728072e-05
```

Como podemos observar, si utilizamos el método bootstrap obtenemos un intervalo medio de $(0.0484, 0.0528)$, mientras que al considerar la distribución asintótica tenemos: $(0.0483, 0.0530)$. Ambos intervalos son muy parecidos y como era de esperar, contienen al verdadero valor del parámetro β_1. Además, si observamos las diferentes desviaciones típicas, tenemos que para todos los casos son considerablemente pequeñas.

Ahora calcularemos el número de intervalos obtenidos, de entre estas 100 réplicas del experimento, que no contienen al valor real 0.05:

```
# Número de fallos mediante bootstrap
sum(intervalo1_boots[,1]>0.05 | intervalo1_boots[,2]<0.05)
[1] 1

# Número de fallos mediante la distribución asintótica
sum(intervalo1_AD[,1]>0.05 | intervalo1_AD[,2]<0.05)
[1] 1
```

Para ambos casos, tenemos que todos sólo un intervalo de los 100 calculados no contiene al verdadero valor del parámetro β_1.

Continuamos analizando el segundo de los ejemplos: cálculo del intervalo de confianza para β_2. En este caso, el valor real que debería estar contenido en el intervalo es 0.01. Calculamos la media y la desviación típica para los extremos de los intervalos.

```
# Segunda componente de beta mediante bootstrap
# Extremo inferior
mean(intervalo2_boots[,1])
0.008815404
sd(intervalo2_boots[,1])
7.780249e-05

# Extremo superior
mean(intervalo2_boots[,2])
0.01301463
sd(intervalo2_boots[,2])
0.0001657755


# Segunda componente de beta mediante distribución asintótica
# Extremo inferior
```

```
mean(intervalo2_AD[,1])
0.008687021
sd(intervalo2_AD[,1])
5.107875e-05

# Extremo superior
mean(intervalo2_AD[,2])
0.01294883
sd(intervalo2_AD[,2])
0.0001635651
```

Por un lado, utilizando el método bootstrap tenemos un intervalo medio de $(0.0088, 0.0130)$; por otro lado, considerando la distribución asintótica obtenemos: $(0.0087, 0.0129)$. Al igual que ocurría con el primer ejemplo, también ambos métodos nos dan resultados muy parecidos. Cabe destacar que, como era de esperar, los dos intervalos de confianza resultantes contienen al verdadero valor del parámetro β_2.

Para completar el estudio, obtendremos el número de veces que el valor real 0.01 no se encuentra dentro de los intervalos:

```
## Número de veces que 0.01 cae fuera del intervalo
sum(intervalo2_boots[,1]>0.01 | intervalo2_boots[,2]<0.01)
[1] 0

## Número de veces que 0.01 cae fuera del intervalo
sum(intervalo2_AD[,1]>0.01 | intervalo2_AD[,2]<0.01)
[1] 0
```

Tanto para el método bootstrap como para la distribución asintótica, el valor de la segunda componente del parámetro β está incluido en todos los intervalos para los 100 experimentos realizados.

A la vista de estos resultados, podemos concluir que el funcionamiento de *plrm.ci* es correcto.

8.4 Aplicación a datos reales

Aplicamos ahora la rutina *plrm.ci* a los mismos datos y modelo que en la Sección 3.4. Concretamente, construiremos intervalos de confianza para cada uno de los parámetros del modelo. Esto se hará tanto en base a la distribución asintótica como a través del procedimiento bootstrap.

Comenzamos construyendo la matriz de datos.

```
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))
```

A continuación aplicamos la función *plrm.ci* para construir intervalos de confianza para la primera componente del parámetro β.

```
b.h <- plrm.gcv(data)$bh.opt
b1 <- b.h[1]

plrm.ci(data, b1=b1, b2=b1, a=c(1,0), CI="all")
```

```
> plrm.ci(data, b1=b1, b2=b1, a=c(1,0), CI="all")
$Bootstrap
      ci_inf    ci_sup p_opt q_opt       b1       b2
1 -0.7146756 0.4647373     0     0 9.785714 9.785714

$AD
      ci_inf    ci_sup p_opt q_opt       b1
1 -0.7409779 0.4574704     0     0 9.785714

$pv.Box.test
 [1] 0.9748213 0.9993648 0.5909427 0.5847114 0.4515939
 [6] 0.3574940 0.4695164 0.5719943 0.6640864 0.4726630

$pv.t.test
[1] 0.9440985
```

Figure 9: Resultados de *plrm.ci* para la primera componente de β

En la Figura 9 podemos observar que los intervalos de confianza por el método bootstrap y por la distribución asintótica son bastante similares, ya que mientras para el bootstrap obtenemos un intervalo de confianza de $(-0.7147, 0.4647)$, para la distribución asintótica tenemos $(-0.7410, 0.4575)$.

Un aspecto a destacar es que la conclusión que obtendríamos analizando estos resultados es que la primera componente de β no es significativa (al 5%) en el modelo, al estar incluido el 0 en ambos intervalos. Es decir, según esto, tendríamos que los cambios en el precio del percebe en Cedeira no influyen de manera significativa en los cambios de las ventas de percebe en la misma localidad. En realidad, esta conclusión podría parecer sorprendente, en el sentido de que una disminución en el precio viene, en general, acompañada de un aumento de ventas, y viceversa. Posiblemente la no significatividad de β_1 venga provocada por la alta variabilidad de los errores del modelo, la cual debiera disminuir a medida que se incorporen a él nuevas variables explicativas con información significativa (en la Sección 10 del trabajo indicamos

algunos ejemplos de tales variables). En cualquier caso, es destacable que la estimación puntual de β_1 fue negativa.

Por otra parte, hacemos el test para la segunda componente de β:

```
plrm.ci(data, b1=b1, b2=b1, a=c(0,1), CI="all")
```

```
> plrm.ci(data, b1=b1, b2=b1, a=c(0,1), CI="all")
$Bootstrap
     ci_inf    ci_sup p_opt q_opt       b1       b2
1 0.3039493 0.5051234     0     0 9.785714 9.785714

$AD
     ci_inf    ci_sup p_opt q_opt       b1
1 0.3056561 0.5186994     0     0 9.785714

$pv.Box.test
 [1] 0.9748213 0.9993648 0.5909427 0.5847114 0.4515939
 [6] 0.3574940 0.4695164 0.5719943 0.6640864 0.4726630

$pv.t.test
[1] 0.9440985
```

Figure 10: Resultados de *plrm.ci* para la segunda componente de β

A la vista de la Figura 10, tenemos que el intervalo de confianza calculado para β_2 por el método bootstrap es de $(0.3039, 0.5051)$ y mediante la distribución asintótica, de $(0.3057, 0.5187)$. En ambos casos se corrobora el signo positivo del parámetro correspondiente, siendo por tanto su interpretanción análoga a la hecha cuando se estimó puntualmente.

9 Programa plrm.ancova

9.1 Objetivo

En este apartado consideramos el problema del análisis de la covarianza, es decir, la comparación del efecto de un conjunto de variables explicativas en la respuesta observada en diferentes grupos. Concretamente, estudiaremos los modelos de regresión parcialmente lineales con $L \geq 2$ grupos y realizaremos dos tipos de contrastes de hipótesis: compararemos los efectos de la parte lineal y los de la parte no paramétrica sobre la respuesta del modelo.

Por lo tanto, dados los L modelos de regresión parcialmente lineales:

$$Y_{l,i} = X_{l,i}^T \beta_l + m_l(T_{l,i}) + \varepsilon_{l,i}, \qquad i = 1, \ldots, n/L; \;\; l = 1, \ldots, L, \tag{49}$$

el programa *plrm.ancova*, a partir de una muestra $(Y_{li}, X_{li1}, \ldots, X_{lip}, T_i)$ con $i = 1, \ldots, n/L$ y $l = 1, \ldots, L$, realiza los siguientes contrastes de hipótesis:

$$\begin{cases} H_{0\beta} : \beta_1 = \cdots = \beta_L \\ H_{1\beta} : \exists \beta_i,\ \beta_i \neq \beta_j,\ \forall j \neq i \end{cases} \tag{50}$$

y

$$\begin{cases} H_{0m} : m_1 = \cdots = m_L \\ H_{1m} : \exists m_i,\ m_i \neq m_j,\ \forall j \neq i \end{cases} \tag{51}$$

9.2 Metodología

Mostraremos los estadísticos para contrastar las hipótesis de $H_{0\beta}$ y H_{0m}, que deben estar basados en los estimadores para β_l y m_l, respectivamente. Para el modelo (49), tenemos que el estimador para la parte paramétrica será el propuesto en Robinson (1988) en [7] y Speckman (1988) en [8]:

$$\hat{\beta}_{l,b} = (\tilde{X}_{l,b}^T \tilde{X}_{l,b})^{-1} \tilde{X}_{l,b}^T \tilde{Y}_{l,b}, \tag{52}$$

y para la parte no paramétrica, Speckman (1988) en [8] considera:

$$\hat{m}_{l,h}(T, \hat{\beta}_{l,b}) = \sum_{i=1}^{n_l} w_{l,h}(T, T_{l,i})(Y_{l,i} - X_{l,i}^T \hat{\beta}_{l,b}). \tag{53}$$

Aneiros-Pérez (2008) [24] propone, bajo ciertas condiciones, el siguiente estadístico para realizar el contraste paramétrico (50)

$$\hat{Q}_{n,\beta} = n(B\hat{\beta}_b)^T (BAB^T)^{-1} (B\hat{\beta}_b) \tag{54}$$

y para el contraste no paramétrico (51):

$$\hat{Q}_{n,m} = \sum_{l=2}^{L} \sum_{s=1}^{l-1} \int (\hat{m}_{l,h}(t, \hat{\beta}_{l,b}) - \hat{m}_{s,h}(t, \hat{\beta}_{s,b}))^2 w_{l,s}(t) dt \tag{55}$$

siendo

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A_l \end{bmatrix}$$

donde A_l es la matriz de varianzas-covarianzas asintótica de $n_l^{1/2}(\hat{\beta}_{l,b} - \beta_l)$ (hemos denotado $n_l = n/L$), y

$$B = \begin{bmatrix} I_p & 0 & \cdots & 0 & -I_p \\ 0 & I_p & \ddots & \vdots & -I_p \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & I_p & -I_p \end{bmatrix}.$$

$w_{l,s}(\cdot)$ es una función de pesos introducida para eliminar (o, al menos, reducir) los problemas con los efectos frontera al estimar las componentes no paramétricas.

Mostraremos los resultados asintóticos de los estadísticos (54) y (55), que son estudiados en Aneiros-Perez (2008) [24] bajo ciertas condiciones.

El Teorema 1 de dicho artículo concluye que, bajo la hipótesis nula $H_{0\beta}$:

$$\hat{Q}_{n,\beta} \xrightarrow{d} \chi^2_{p(L-1)}, \tag{56}$$

donde $\chi^2_{p(L-1)}$ denota la distribución Chi-cuadrado con $p(L-1)$ grados de libertad.

Por otra parte, utilizando el Teorema 2 y asumiendo diseño fijo y equiespaciado en $[0,1]$ para T, bajo la hipótesis nula H_{0m} tenemos que:

$$\frac{\sqrt{n^2h}}{\sigma_Q}\left(\hat{Q}_{n,m} - \frac{\sum_{j=1}^{L}\left(\Gamma_j \sum_{l=1,l\neq j}^{L}\int \frac{w_{j,l}}{\pi_j}\right)\int K^2}{nh}\right) \xrightarrow{d} N(0,1), \tag{57}$$

siendo

$$\sigma_Q^2 = 2\int (K*K)^2 \left(\sum_{j=1}^{L}\Gamma_j^2 \int\left(\sum_{l=1,l\neq j}^{L}\frac{w_{j,l}}{\pi_j}\right)^2 + \sum_{j=1}^{L}\sum_{l=1,l\neq j}^{L}\Gamma_j\Gamma_l\int\frac{w_{j,l}^2}{\pi_j\pi_l}\right),$$

donde $\Gamma_j = \sum_{k=-\infty}^{\infty} r_{\varepsilon_j}(k)$, $K*K$ denota la convolución de K consigo misma y $\pi_j = \frac{n_j}{n} = \frac{1}{L}$.

Las distribuciones asintóticas (56) y (57) generalizan las basadas en regresión lineal y regresión no paramétrica, respectivamente, propuestas por Seber (1977) en [1] y Vilar-Fernández y González-Manteiga (2004) en [21].

Denotando $\hat{Q}^*_{n,\beta}$ al estadístico obtenido cambiando A en $\hat{Q}_{n,\beta}$ por un estimador consistente $\hat{A}$, Aneiros-Pérez (2008) [24] propone rechazar $H_{0\beta}$ para un nivel de significación α si:

$$\hat{Q}^*_{n,\beta} > \Psi^{-1}(1-\alpha) \tag{58}$$

y rechazar H_{0m} si:

$$\hat{Q}_{n,m} > \frac{\hat{\sigma}_Q}{\sqrt{n^2h}}\Phi(1-\alpha) + \frac{\sum_{j=1}^{L}\left(\hat{\Gamma}_j \sum_{l=1,l\neq j}^{L}\int \frac{w_{j,l}}{\pi_j}\right)\int K^2}{nh}, \tag{59}$$

donde Ψ y Φ son las funciones de distribución acumuladas de la $\chi^2_{p(L-1)}$ y de la Normal estándar, respectivamente. En ambos casos, los estimadores $\hat{A}$ y $\hat{\Gamma}_j$ (y por tanto, $\hat{\sigma}_Q$) están basados en modelos ARMA ajustados a los residuos.

9.3 Aplicación a datos simulados

Para mostrar el correcto funcionamiento de la rutina *plrm.ancova*, preparamos un array de datos simulados de la siguiente forma: comenzamos construyendo una matriz de forma análoga a la Sección 3.3. Esa matriz la metemos en la primera "capa" del array. Por otra parte, generamos otra matriz de forma similar (utilizando los mismos β y m), con las variables explicativas X generadas a partir de una N(1,2) y con ε generada por un proceso MA(1). Incluimos dicha matriz en la segunda "capa" de nuestro array. De esta forma, al hacer el contraste sobre nuestro conjunto de datos, deberíamos obtener que los modelos sí se pueden considerar iguales (al haber sido ambos construidos con la misma parte paramétrica y no paramétrica).

Además, haremos el contraste para el caso contrario. Es decir, en la segunda "capa" del array incluiremos una matriz de datos generada con $\beta = (0.05, 0.02)$ y m una función del tipo $0.25 * t * (1 - t) * 0.75$. Por lo tanto, al aplicar la función sería lógico obtener que ni la parte paramétrica ni la parte no paramétrica se pueden considerar iguales en ambos modelos.

Cabe destacar que utilizaremos el método de validación cruzada generalizada para calcular la ventana óptima para estimar β y m. Comenzamos con el **primer ejemplo:** H_0 cierta.

En la matriz `contraste` almacenaremos los p-valores correspondientes. Repetiremos este proceso 1000 veces y luego calcularemos la proporción de fallos:

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
b.seq <- seq(0.01, 0.25, length=50)

contraste <- matrix(NA, 1000, 2)
colnames(contraste) <- c("Paramétrico","No paramétrico")
```

```
seed <- 1234
set.seed(seed)

for (i in 1:1000) {

  cat(i, "/", 1000, ", ")

  x1 <- matrix(rnorm(200,0,1), nrow=n)
  sum1 <- x1%*%beta
  epsilon1 <- arima.sim(list(order=c(1,0,0),ar=0.7),sd=0.01,n=n)
  y1 <-  sum1 + f + epsilon1
  data1 <- cbind(y1,x1)

  x2 <- matrix(rnorm(200,1,2), nrow=n)
  sum2 <- x2%*%beta
  epsilon2 <- arima.sim(list(order=c(0,0,1),ma=0.5),sd=0.02,n=n)
  y2 <- sum2 + f + epsilon2
  data2 <- cbind(y2,x2)

  data_eq <- array(c(data1,data2), c(n,3,2))

  ## Ejemplo 1: H0 cierta
  b.h <- plrm.gcv(cbind(data1, t=t, b.seq=b.seq))$bh.opt
  result <- plrm.ancova(data=data_eq, t=t, b.seq=b.h[1],
  h.seq=b.h[2], time.series=TRUE)

  contraste[i,1] <- result$parametric.test$p.v
  contraste[i,2] <- result$nonparametric.test$p.v
}
```

Ahora calcularemos, para un α prefijada, la proporción de fallos del contraste (rechazos de H_0 siendo cierta); es decir, estaríamos obteniendo la proporción de error de tipo 1, o lo que es lo mismo, el nivel de significación empírico.

```
alpha <- 0.05
sum(contraste[,1]<alpha)/1000
0.034
sum(contraste[,2]<alpha)/1000
0.041
```

Para el contraste paramétrico obtenemos un nivel de significación empírico de 0.034 y para el contraste no paramétrico, de 0.041.

Continuamos con el **segundo ejemplo**: H_0 falsa.

```
# Generamos los datos
n <- 100
t <- ((1:n)-0.5)/n
beta1 <- c(0.05, 0.01)
beta2 <- c(0.05, 0.02)
m1 <- function(t) {0.25*t*(1-t)}
m2 <- function(t) {0.25*t*(1-t)*0.75}
f1 <- m1(t)
f2 <- m2(t)
b.seq <- seq(0.01, 0.25, length=50)

contraste <- matrix(NA, 1000, 2)
colnames(contraste) <- c("Paramétrico","No paramétrico")

seed <- 1234
set.seed(seed)

for (i in 1:1000) {

  cat(i, "/", 1000, ", ")

  x1 <- matrix(rnorm(200,0,1), nrow=n)
  sum1 <- x1%*%beta1
  epsilon1 <- arima.sim(list(order=c(1,0,0),ar=0.7),sd=0.01,n=n)
  y1 <-  sum1 + f1 + epsilon1
  data1 <- cbind(y1,x1)

  x2 <- matrix(rnorm(200,1,2), nrow=n)
  sum2 <- x2%*%beta2
  epsilon2 <- arima.sim(list(order=c(0,0,1),ma=0.5),sd=0.02,n=n)
  y2 <-  sum2 + f2 + epsilon2
  data2 <- cbind(y2,x2)

  data_neq <- array(c(data1,data2), c(n,3,2))

  ## Ejemplo 2: H0 falsa
  b.h <- plrm.gcv(cbind(data1,t=t, b.seq=b.seq))$bh.opt
  result <- plrm.ancova(data=data_neq, t=t, b.seq=b.h[1],
  h.seq=b.h[2], time.series=TRUE)

  contraste[i,1] <- result$parametric.test$p.v
  contraste[i,2] <- result$nonparametric.test$p.v
}
```

Para este caso, calcularemos, para un α prefijado, la proporción de error de tipo 2; es decir, obtendremos la proporción de veces que el contraste acepta H_0 siendo H_1 cierta.

```
alpha <- 0.05
sum(contraste[,1]>alpha)/1000
0
sum(contraste[,2]>alpha)/1000
0.587
```

Es importante destacar que por un lado, la parte paramétrica en el modelo 1 es de la forma: $\beta_1 = (0.05, 0.01)$ y en el modelo 2 es: $\beta_2 = (0.05, 0.02)$; por lo que no existen grandes diferencias entre ambas y aún así nuestra función no las considera iguales en ninguna de las 1000 réplicas del experimento.

Por otro lado, la función m en el modelo 1 es del tipo: $0.25 * t * (1 - t)$, mientras que en el modelo 2 es: $0.25 * t * (1 - t) * 0.75$. Para este caso, las funciones son también bastante parecidas y aquí el porcentaje de error ya es bastante más elevado que para el caso anterior. Para intentar reducir este error, consideramos una función en el modelo 2 que se diferencie de forma más clara a la del modelo 1, por ejemplo, $0.25 * t * (1 - t) * 0.5$. Aplicamos el contraste y conseguimos una proporción de fallos mucho más pequeña:

```
[1] 0.047
```

Analizando estos resultados, podemos concluir que la función *plrm.ancova* tiene un comportamiento satisfactorio.

9.4 Aplicación a datos reales

Aplicamos ahora la rutina *plrm.ancova* a los mismos datos que en la Sección 3.4. Además, utilizaremos los datos de *barnacles2*, que transformamos análogamente a lo hecho con *barnacles1*.

Por lo tanto, primero preparamos los datos tal y como los requiere el argumento `data` en *plrm.ancova*. Tendremos un modelo (digamos modelo 1) con los datos de *barnacles1* y otro modelo (digamos modelo 2) con los datos de *barnacles2*, y consideramos que ambos pertenecen a un modelo del tipo (15). Construimos un array con un modelo en cada una de las dimensiones e incluimos T en otra variable fuera del array.

```
data(barnacles1)
data <- as.matrix(barnacles1)
```

```
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

data(barnacles2)
data2 <- as.matrix(barnacles2)
data2 <- diff(data2, 12)
data2 <- cbind(data2,1:nrow(data2))

data3 <- array(0, c(nrow(data),ncol(data)-1,2))
data3[,,1] <- data[,-4]
data3[,,2] <- data2[,-4]
t <- data[,4]
```

Una vez preparados los datos, aplicamos la función *plrm.ancova*:

```
plrm.ancova(data=data3, t=t)
```

Que nos da como resultado:

```
> plrm.ancova(data=data3, t=t)
$parametric.test
      b         Q       p.v
1   5.4 0.2719952 0.8728447
2   7.8 0.1770573 0.9152769
3  10.2 0.1190920 0.9421922
4  12.6 0.2100892 0.9002843
5  15.0 0.3848950 0.8249376
6  17.4 0.4527341 0.7974253
7  19.8 0.4640470 0.7929275
8  22.2 0.4131193 0.8133777
9  24.6 0.3474029 0.8405478
10 27.0 0.2810292 0.8689110

$nonparametric.test
      b    h           Q Q.normalised       p.v
1   5.4  5.4 0.034817527   1.05522799 0.1456605
2   7.8  7.8 0.021811812   0.59056060 0.2774074
3  10.2 10.2 0.013859318   0.11207332 0.4553826
4  12.6 12.6 0.010327076  -0.04136396 0.5164971
5  15.0 15.0 0.008518771  -0.06502907 0.5259246
6  17.4 17.4 0.006861594  -0.15066958 0.5598818
7  19.8 19.8 0.005381172  -0.27082831 0.6067385
8  22.2 22.2 0.004107490  -0.40212613 0.6562044
9  24.6 24.6 0.003195523  -0.49583739 0.6899954
10 27.0 27.0 0.002604029  -0.54500429 0.7071247
```

Figure 11: Resultados de *plrm.ancova* para datos reales

Nótese que, al no haber especificado las ventanas, el contraste se ejecuta para distintos valores del par `(b,h)`. Observando la Figura 11, tenemos que

tanto para la parte paramétrica como para la no paramétrica, los p-valores resultantes son muy altos con las diferentes ventanas.

Por lo tanto, podemos concluir que el cambio anual (en proporción) en el número de ventas de percebes en Cedeira se ve influenciado por los correspondientes cambios en el precio de los mismos y en el número de ventas en una localidad cercana, de la misma forma que el cambio anual (en proporción) en el número de ventas de percebes en Cangas se ve influenciado por los correspondientes cambios en el precio de los mismos y en el número de ventas en Baiona.

10 Trabajo futuro

Desde el punto de vista de la metodología del paquete, tenemos previsto incorporar las rutinas *plrm.vs* y *par.vs* para la selección de variables que entran en los modelos de manera lineal. Para eso, seguiremos los artículos de Huang y Xie (2007) [23] para el caso del modelo lineal, y de Xie y Huang (2009) [25], para el modelo parcialmente lineal.

Por otra parte, desde el punto de vista de los datos reales, pretendemos incluir en las matrices *barnacles1* y *barnacles2* los datos de *fuerza* y *dirección del viento*. La inclusión de estas variables en el modelo posiblemente mejorará el ajuste, disminuirá la variabilidad y aumentará la potencia de los contrastes.

Además, consideramos escribir un paper detallando los métodos implementados en el paquete *PLRModels*.

Bibliografía

[1] G.A.F. Seber, 1977
Linear Regression Analysis, New York, Willey

[2] P. Craven y G. Wahba, 1979
Smoothing noisy data with spline functions, Numer. Math, Vol. 31, pp. 377 - 403

[3] G. Golub, M. Heath y G. Wahba, 1979
Generalized cross validation as a method for choosing a good ridge parameter, Technometrics, Vol. 21, pp. 215 - 223

[4] J. Rice, 1984
Bandwidth choice for nonparametric regression, Ann. Statist. , Vol. 12, No 4, pp. 1215 - 1230

[5] W. Hardle y J.S. Marron, 1985
Optimal bandwidth selection in nonparametric regression function estimation, Ann. Statist, Vol. 13, No 4, pp. 1465 - 1481

[6] R.F. Engle, C.W.J. Granger, J. Rice y A. Weiss, 1986
Nonparametric estimates of the relation between weather and electricity sales, Journal of the american Statistical Association, Vol. 81, pp. 310 - 320

[7] P. Robinson, 1988
Root-n-consistent semiparametric regression, Econometrica, Vol. 56, pp. 931 - 954

[8] P. Speckman, 1988
Kernel smoothing in partial linear models, Journal of the Royal Statistical Society, Series B, Vol. 50, No 3, pp. 413 - 436

[9] S. Csorgo y J. Mielniczuk, 1995
Nonparametric regression under long-range dependent normal errors, Ann. Statist., Vol. 23, pp. 1000 - 1014

[10] J.Gao, 1995
Asymptotic theory for partly linear models, Comm. Statist Theory Methods, Vol. 24, pp. 1985 - 2009

[11] R.S. Deo, 1997
Nonparametric regression with long-memory errors, Statist. Probab. Lett, Vol. 33, pp. 89 - 94

[12] J. Gao, 1997
Adaptive Parametric Test in a Semiparametric Regression Model, Comm. Statist. Theory Methods, Vol. 26, pp. 787 - 800

[13] J. Beran y Y. Feng, 1998
Root-n-consistent estimation in partial linear models with long-memory errors, Scand. J. Statist, Vol. 25, pp. 345 - 357

[14] Schmalensee y Stocker, 1999
Household gasoline demand in the United States, Econometrica, Vol. 67, pp. 645 - 662

[15] H. Liang, W. Hardle y V. Sommerfeld, 2000
Bootstrap Approximation in a Partially Linear Regression Model, Journal of Statistical Planning and Inference, Vol. 91, pp. 413 - 426

[16] W, Hardle, H. Liang y J. Gao, 2000
Partially Linear Models, Physica Verlag

[17] G. Aneiros-Pérez y A. Quintela-del-Río, 2001
Modified Cross-Validation in Semiparametric regression models with dependent errors, Comm. Statist Theory Methods, Vol. 30, pp. 289 - 307

[18] J.T. Gao, 2001
Asymptotic theory for partly linear models, Comm. Statist Theory Methods, Vol. 24, pp. 1985 - 2009

[19] G. Aneiros-Pérez y W. González-Manteiga, 2003
Testing in Partial Linear Regression Models with Dependent Errors, J. Nonparametr. Statist., Vol. 15, pp. 93 - 111

[20] G. Aneiros-Pérez, W. González-Manteiga y P. Vieu, 2004
Estimation and Testing in a Partial Linear Regression under Long-memory Dependence, Bernoulli, Vol. 10, pp. 49 - 78

[21] J.M. Vilar-Fernández y W. González-Manteiga, 2004
Nonparametric comparison of curves with dependent errors, Statistics , Vol. 38, pp. 81 - 99

[22] J. You y X. Zhou, 2005
Bootstrap of a Semiparametric Partially Linear Model with Autoregressive Errors, Statistica Sinica, Vol. 15, pp. 117 - 133

[23] J. Huang y H. Xie, 2007
Asymptotic oracle properties of SCAD-penalized least squares estimators, Lecture Notes-Monograph Series, Vol. 55, pp. 149 - 166

[24] G. Aneiros-Pérez, 2008
Semi-parametric analysis of covariance under dependence conditions within each group, Aust. N. Z. J. Stat, Vol. 50(2), pp. 97 - 123

[25] H. Xie y J. Huang, 2009
SCAD-penalized regression in high-dimensional partially linear models, Ann. Statist., Vol. 37, No 2, pp. 673 - 696

[26] G. Aneiros-Pérez y P. Vieu, 2013
Testing Linearity in Semi-parametric Functional Data Analysis, Comput. Stat., Vol. 28, pp. 413 - 434

[27] G. Aneiros-Pérez y A. López-Cheda, (2014)
PLRModels: Statistical inference in partial linear regression models. R package version 1.1. `http://CRAN.R-project.org/package=PLRModels`

Anexo

A PLRModels

Package 'PLRModels'

January 1, 2014

Type Package

Title Statistical inference in partial linear regression models

Version 1.1

Date 2014-01-01

Author German Aneiros Perez and Ana Lopez Cheda

Maintainer German Aneiros Perez <ganeiros@udc.es>

Description

This package provides statistical inference tools applied to Partial Linear Regression (PLR) models. Specifically, point estimation, confidence intervals estimation, bandwidth selection, goodness-of-fit tests and analysis of covariance are considered.
Kernel-based methods, combined with ordinary least squares estimation, are used and time series errors are allowed. In addition, these techniques are also implemented for both parametric (linear) and nonparametric regression models.

License GPL

R topics documented:

`PLRModels-package` *Statistical inference in partial linear regression models*

Description

This package provides statistical inference tools applied to Partial Linear Regression (PLR) models. Specifically, point estimation, confidence intervals estimation, bandwidth selection, goodness-of-fit tests and analysis of covariance are considered. Kernel-based methods, combined with ordinary least squares estimation, are used and time series errors are allowed. In addition, these techniques are also implemented for both parametric (linear) and nonparametric regression models.

Details

Package:	PLRModels
Type:	Package
Version:	1.1
Date:	2014-01-01
License:	GPL

The most important functions are those directly related with the PLR models; that is, `plrm.gcv`, `plrm.cv`, `plrm.beta`, `plrm.est`, `plrm.gof`, `plrm.ancova` and `plrm.ci`. Although the other functions included in the package are auxiliary ones, they can be used independiently.

Author(s)

Authors: German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

Maintainer: German Aneiros Perez <ganeiros@udc.es>

`barnacles1` *Sales of barnacles in Cedeira*

Description

Information about sales and prices of barnacles in two galician towns for each month from 2004 to 2013. The data have been transformed using the logarithm function.

Usage

```
data(barnacles1)
```

Format

A matrix containing 3 columns:

`barnacles1[, 1]` contains the number of sales (in kg) of barnacles in Cedeira's fish market;

`barnacles1[, 2]` contains the prices (in euro/kg) of the barnacles in Cedeira's fish market;

`barnacles1[, 3]` contains the number of sales (in kg) of barnacles in Carino's fish market.

Source

http://www.pescadegalicia.com/

`barnacles2` *Sales of barnacles in Cangas*

Description

Information about sales and prices of barnacles in two galician towns for each month from 2004 to 2013. The data have been transformed using the logarithm function.

Usage

```
data(barnacles2)
```

Format

A matrix containing 3 columns:

`barnacles1[, 1]` contains the number of sales (in kg) of barnacles in Cangas' fish market;

`barnacles1[, 2]` contains the prices (in euro/kg) of the barnacles in Cangas' fish market;

`barnacles1[, 3]` contains the number of sales (in kg) of barnacles in Baiona's fish market.

Source

http://www.pescadegalicia.com/

`np.ancova` *Nonparametric analysis of covariance*

Description

This routine tests the equality of L nonparametric regression curves $(m_1, ..., m_L)$ from samples $(Y_{ki}, t_i) : i = 1, ..., n, k = 1, ..., L$, where:

$$Y_{ki} = m_k(t_i) + \epsilon_{ki}.$$

The unknown functions m_k are smooth, fixed equally spaced design is considered, and the random errors, ϵ_{ki}, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : m_1 = ... = m_L$, derives from a Cramer-von-Mises-type functional based on different distances between nonparametric estimators of the regression functions.

Usage

```
np.ancova(data = data, h.seq = NULL, w = NULL, estimator = "NW",
kernel = "quadratic", time.series = FALSE, Tau.eps = NULL,
h0 = NULL, lag.max = 50, p.max = 3, q.max = 3, ic = "BIC",
num.lb = 10, alpha = 0.05)
```

Arguments

data	data[, k] contains the values of the response variable, Y_k, for each model k ($k = 1, ..., L$); data[, L+1] contains the values of the explanatory (common) variable, t, for each model k ($k = 1, ..., L$).
h.seq	the statistic test is performed using each bandwidth in the vector h.seq (the same bandwidth is used to estimate all the regression functions). If NULL (the default), 10 equidistant values between 0 and the first half of the range of t_i are considered.
w	support interval of the weigth function in the test statistic. If NULL (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
time.series	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
Tau.eps	Tau.eps[k] contains the sum of autocovariances associated to the random errors of the regression model k ($k = 1, ..., L$). If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted nonparametric regression model and, then, it obtains the sum of the autocovariances of such ARMA model.
h0	if Tau.eps=NULL, h0 contains the pilot bandwidth used for obtaining the residuals to construct the default for Tau.eps. If NULL (the default), a quarter of the range of t_i is considered.
lag.max	if Tau.eps=NULL, lag.max contains the maximum delay used to construct the default for Tau.eps. The default is 50.
p.max	if Tau.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Tau.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Tau.eps=NULL, ic contains the information criterion used to suggest the ARMA models. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Tau.eps=NULL, it checks the suitability of the selected ARMA models according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Tau.eps=NULL, alpha contains the significance level which the ARMA models are checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

If Tau.eps=NULL and the routine is not able to suggest an approximation for Tau.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Tau.eps, the procedures suggested in Muller and Stadmuller (1988) and Herrmann *et al.* (1992) can be followed.

For more details, see Vilar-Fernandez and Gonzalez-Manteiga (2004).

Value

A list with a dataframe containing:

h.seq	sequence of bandwidths used in the test statistic.
Q.m	values of the test statistic (one for each bandwidth in h.seq).
Q.m.normalised	normalised value of Q.m.
p.value	p-values of the corresponding statistic tests (one for each bandwidth in h.seq).

Moreover, if data is a time series and Tau.eps is not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Dette, H. and Neumeyer, N. (2001) Nonparametric analysis of covariance. *Ann. Statist.* **29**, no. 5, 1361-1400.

Herrmann, E., Gasser, T. and Kneip, A. (1992) Choice of bandwidth for kernel regression when residuals are correlated. *Biometrika* **79**, 783-795

Muller, H.G. and Stadmuller, U. (1988) Detecting dependencies in smooth regression models. *Biometrika* **75**, 639-650

Vilar-Fernandez, J.M. and Gonzalez-Manteiga, W. (2004) Nonparametric comparison of curves with dependent errors. *Statistics* **38**, 81-99.

See Also

Other related functions are np.est, par.ancova and plrm.ancova.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

data2 <- matrix(10,120,2)
data(barnacles2)
barnacles2 <- as.matrix(barnacles2)
data2[,1] <- barnacles2[,1]
data2 <- diff(data2, 12)
data2[,2] <- 1:nrow(data2)

data3 <- matrix(0, nrow(data),ncol(data)+1)
```

```
data3[,1] <- data[,1]
data3[,2:3] <- data2

np.ancova(data=data3)



# EXAMPLE 2: SIMULATED DATA
## Example 2.1: dependent data: true null hypothesis

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m1 <- function(t) {0.25*t*(1-t)}
f <- m1(t)

epsilon1 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y1 <-  f + epsilon1

epsilon2 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y2 <- f + epsilon2

data_eq <- cbind(y1, y2, t)

# We apply the test
np.ancova(data_eq, time.series=TRUE)


## Example 2.2: dependent data: false null hypothesis
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m3 <- function(t) {0.25*t*(1-t)}
m4 <- function(t) {0.25*t*(1-t)*0.75}

f3 <- m3(t)
epsilon3 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y3 <-  f3 + epsilon3

f4 <- m4(t)
epsilon4 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y4 <-  f4 + epsilon4

data_neq<- cbind(y3, y4, t)

# We apply the test
np.ancova(data_neq, time.series=TRUE)
```

---

np.cv *Cross-validation bandwidth selection in nonparametric regression models*

Description

From a sample $(Y_i, t_i) : i = 1, ..., n$, this routine computes, for each l_n considered, an optimal bandwidth for estimating m in the regression model

$$Y_i = m(t_i) + \epsilon_i.$$

The regression function, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series. The optimal bandwidth is selected by means of the leave-$(2l_n + 1)$-out cross-validation procedure. Kernel smoothing is used.

Usage

```
np.cv(data = data, h.seq = NULL, num.h = 50, w = NULL, num.ln = 1,
ln.0 = 0, step.ln = 2, estimator = "NW", kernel = "quadratic")
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2]` contains the values of the explanatory variable, t.
`h.seq`	sequence of considered bandwidths in the CV function. If `NULL` (the default), `num.h` equidistant values between zero and a quarter of the range of t_i are considered.
`num.h`	number of values used to build the sequence of considered bandwidths. If `h.seq` is not `NULL`, `num.h=length(h.seq)`. Otherwise, the default is 50.
`w`	support interval of the weigth function in the CV function. If `NULL` (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
`num.ln`	number of values for l_n: $2l_n+1$ observations around each point t_i are eliminated to estimate $m(t_i)$ in the CV function. The default is 1.
`ln.0`	minimum value for l_n. The default is 0.
`step.ln`	distance between two consecutives values of l_n. The default is 2.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the CV function to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

For more details, see Chu and Marron (1991).

Value

`h.opt`	dataframe containing, for each `ln` considered, the selected value for the bandwidth.
`CV.opt`	`CV.opt[k]` is the minimum value of the CV function when de k-th value of `ln` is considered.
`CV`	matrix containing the values of the CV function for each bandwidth and `ln` considered.
`w`	support interval of the weigth function in the CV function.
`h.seq`	sequence of considered bandwidths in the CV function.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Chu, C-K and Marron, J.S. (1991) Comparison of two bandwidth selectors with dependent errors. *The Annals of Statistics* **19**, 1906-1918.

See Also

Other related functions are: np.est, np.gcv, plrm.est, plrm.gcv and plrm.cv.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

aux <- np.cv(data, ln.0=1,step.ln=1, num.ln=2)
aux$h.opt
plot.ts(aux$CV)

par(mfrow=c(2,1))
plot(aux$h.seq,aux$CV[,1], xlab="h", ylab="CV", type="l", main="ln=1")
plot(aux$h.seq,aux$CV[,2], xlab="h", ylab="CV", type="l", main="ln=2")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

epsilon <- rnorm(n, 0, 0.01)
y <-  f + epsilon
data_ind <- matrix(c(y,t),nrow=100)

# We apply the function
a <-np.cv(data_ind)
a$CV.opt

CV <- a$CV
h <- a$h.seq
plot(h,CV,type="l")
```

`np.est` *Nonparametric estimate of the regression function*

Description

This routine computes estimates for $m(newt_j)$ $(j = 1, ..., J)$ from a sample $(Y_i, t_i) : i = 1, ..., n$, where:

$$Y_i = m(t_i) + \epsilon_i.$$

The regression function, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series. Kernel smoothing is used.

Usage

```
np.est(data = data, h.seq = NULL, newt = NULL,
estimator = "NW", kernel = "quadratic")
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2]` contains the values of the explanatory variable, t.
`h.seq`	the considered bandwidths. If `NULL` (the default), only one bandwidth, selected by means of the cross-validation procedure, is used.
`newt`	values of the explanatory variable where the estimates are obtained. If NULL (the default), the considered values will be the values of `data[,2]`.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

See Fan and Gijbels (1996) and Francisco-Fernandez and Vilar-Fernandez (2001).

Value

YHAT: a length(`newt`) x length(`h.seq`) matrix containing the estimates for $m(newt_j)$

$(j = 1, ...,$length(`newt`)) using the different bandwidths in `h.seq`.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Fan, J. and Gijbels, I. (1996) *Local Polynomial Modelling and its Applications*. Chapman and Hall, London.

Francisco-Fernandez, M. and Vilar-Fernandez, J. M. (2001) Local polynomial regression estimation with correlated errors. *Comm. Statist. Theory Methods* **30**, 1271-1293.

See Also

Other related functions are: np.gcv, np.cv, plrm.est, plrm.gcv and plrm.cv.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

aux <- np.gcv(data)
h <- aux$h.opt
ajuste <- np.est(data=data, h=h)
plot(data[,2], ajuste, type="l", xlab="t", ylab="m(t)")
plot(data[,1], ajuste, xlab="y", ylab="y.hat", main="y.hat vs y")
abline(0,1)
residuos <- data[,1] - ajuste
mean(residuos^2)/var(data[,1])



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

epsilon <- rnorm(n, 0, 0.01)
y <-  f + epsilon
data_ind <- matrix(c(y,t),nrow=100)

# We estimate the nonparametric component of the PLR model
# (CV bandwidth)
est <- np.est(data_ind)
plot(t, est, type="l", lty=2, ylab="")
points(t, 0.25*t*(1-t), type="l")
legend(x="topleft", legend = c("m", "m hat"), col=c("black", "black"), lty=c(1,2))
```

---

np.gcv *Generalized cross-validation bandwidth selection in nonparametric regression models*

Description

From a sample $(Y_i, t_i) : i = 1, ..., n$, this routine computes an optimal bandwidth for estimating m in the regression model

$$Y_i = m(t_i) + \epsilon_i.$$

The regression function, m, is a smooth but unknown function. The optimal bandwidth is selected by means of the generalized cross-validation procedure. Kernel smoothing is used.

Usage

```
np.gcv(data = data, h.seq=NULL, num.h = 50, estimator = "NW",
kernel = "quadratic")
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2]` contains the values of the explanatory variable, t.
`h.seq`	sequence of considered bandwidths in the GCV function. If `NULL` (the default), `num.h` equidistant values between zero and a quarter of the range of t_i are considered.
`num.h`	number of values used to build the sequence of considered bandwidths. If `h.seq` is not `NULL`, `num.h=length(h.seq)`. Otherwise, the default is 50.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

See Craven and Wahba (1979) and Rice (1984).

Value

`h.opt`	selected value for the bandwidth.
`GCV.opt`	minimum value of the GCV function.
`GCV`	vector containing the values of the GCV function for each considered bandwidth.
`h.seq`	sequence of considered bandwidths in the GCV function.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Craven, P. and Wahba, G. (1979) Smoothing noisy data with spline functions. *Numer. Math.* **31**, 377-403.

Rice, J. (1984) Bandwidth choice for nonparametric regression. *Ann. Statist.* **12**, 1215-1230.

See Also

Other related functions are: `np.est`, `np.cv`, `plrm.est`, `plrm.gcv` and `plrm.cv`.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

aux <- np.gcv(data)
aux$h.opt
plot(aux$h.seq, aux$GCV, xlab="h", ylab="GCV", type="l")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

epsilon <- rnorm(n, 0, 0.01)
y <-  f + epsilon
data_ind <- matrix(c(y,t),nrow=100)

# We apply the function
a <-np.gcv(data_ind)
a$GCV.opt

GCV <- a$GCV
h <- a$h.seq
plot(h, GCV, type="l")
```

`np.gof` *Goodness-of-Fit tests in nonparametric regression models*

Description

This routine tests the equality of a nonparametric regression curve, m, and a given function, m_0, from a sample $(Y_i, t_i) : i = 1, ..., n$, where:

$$Y_i = m(t_i) + \epsilon_i.$$

The unknown function m is smooth, fixed equally spaced design is considered, and the random errors, ϵ_i, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : m = m_0$, derives from a Cramer-von-Mises-type functional distance between a nonparametric estimator of m and m_0.

Usage

```
np.gof(data = data, m0 = NULL, h.seq = NULL, w = NULL,
estimator = "NW", kernel = "quadratic", time.series = FALSE,
Tau.eps = NULL, h0 = NULL, lag.max = 50, p.max = 3,
q.max = 3, ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2] contains the values of the explanatory variable, t.
m0	the considered function in the null hypothesis. If NULL (the default), the zero function is considered.
h.seq	the statistic test is performed using each bandwidth in the vector h.seq. If NULL (the default), 10 equidistant values between zero and a quarter of the range of t_i are considered.
w	support interval of the weigth function in the test statistic. If NULL (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
time.series	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
Tau.eps	it contains the sum of autocovariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted nonparametric regression model and, then, it obtains the sum of the autocovariances of such ARMA model.
h0	if Tau.eps=NULL, h0 contains the pilot bandwidth used for obtaining the residuals to construct the default for Tau.eps. If NULL (the default), a quarter of the range of t_i is considered.
lag.max	if Tau.eps=NULL, lag.max contains the maximum delay used to construct the default for Tau.eps. The default is 50.
p.max	if Tau.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Tau.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Tau.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Tau.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Tau.eps=NULL, alpha contains the significance level which the ARMA model is checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

If `Tau.eps=NULL` and the routine is not able to suggest an approximation for `Tau.eps`, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct `Tau.eps`, the procedures suggested in Muller and Stadmuller (1988) and Herrmann *et al.* (1992) can be followed.

The implemented statistic test particularizes that one in Gonzalez Manteiga and Vilar Fernandez (1995) to the case where the considered class in the null hypothesis has only one element.

Value

A list with a dataframe containing:

`h.seq`	sequence of bandwidths used in the test statistic.
`Q.m`	values of the test statistic (one for each bandwidth in `h.seq`).
`Q.m.normalised`	normalised value of Q.m.
`p.value`	p-values of the corresponding statistic tests (one for each bandwidth in `h.seq`).

Moreover, if `data` is a time series and `Tau.eps` is not especified:

`pv.Box.test`	p-values of the Ljung-Box test for the model fitted to the residuals.
`pv.t.test`	p-values of the t.test for the model fitted to the residuals.
`ar.ma`	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Biedermann, S. and Dette, H. (2000) Testing linearity of regression models with dependent errors by kernel based methods. *Test* **9**, 417-438.

Gonzalez-Manteiga, W. and Aneiros-Perez, G. (2003) Testing in partial linear regression models with dependent errors. *J. Nonparametr. Statist.* **15**, 93-111.

Gonzalez-Manteiga, W. and Cao, R. (1993) Testing the hypothesis of a general linear model using nonparametric regression estimation. *Test* **2**, 161-188.

Gonzalez Manteiga, W. and Vilar Fernandez, J. M. (1995) Testing linear regression models using non-parametric regression estimators when errors are non-independent. *Comput. Statist. Data Anal.* **20**, 521-541.

Herrmann, E., Gasser, T. and Kneip, A. (1992) Choice of bandwidth for kernel regression when residuals are correlated. *Biometrika* **79**, 783-795

Muller, H.G. and Stadmuller, U. (1988) Detecting dependencies in smooth regression models. *Biometrika* **75**, 639-650

See Also

Other related functions are `np.est`, `par.gof` and `plrm.gof`.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

np.gof(data)



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
f.function <- function(u) {0.25*u*(1-u)}

epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  f + epsilon
data <- cbind(y,t)

## Example 2a.1: true null hypothesis
np.gof(data, m0=f.function, time.series=TRUE)

## Example 2a.2: false null hypothesis
np.gof(data, time.series=TRUE)
```

`par.ancova` *Parametric analysis of covariance (based on linear models)*

Description

This routine tests the equality of L vector coefficients, $(\beta_1, ..., \beta_L)$, from samples $(Y_{ki}, X_{ki1}, ..., X_{kip})$: $i = 1, ..., n$, $k = 1, ..., L$, where:

$$\beta_k = (\beta_{k1}, ..., \beta_{kp})$$

is an unknown vector parameter and

$$Y_{ki} = X_{ki1} * \beta_{k1} + ... + X_{kip} * \beta_{kp} + \epsilon_{ki}.$$

The random errors, ϵ_{ki}, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : \beta_1 = ... = \beta_L$, derives from the asymptotic normality of the ordinary least squares estimator of β_k $(k = 1, ..., L)$, this result giving a χ^2-test.

Usage

```
par.ancova(data = data, time.series = FALSE, Var.Cov.eps = NULL,
p.max = 3, q.max = 3, ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

data	data[, 1, k] contains the values of the response variable, Y_k, for each model k ($k = 1, ..., L$); data[, 2:(p+1), k] contains the values of the explanatory variables, $X_{k1}, ..., X_{kp}$, for each model k ($k = 1, ..., L$).
time.series	it denotes whether the data is independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
Var.Cov.eps	Var.Cov.eps[, , k] contains the n x n matrix of variances-covariances associated to the random errors of the regression model k ($k = 1, ..., L$). If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted linear regression model and, then, it obtains the var-cov matrix of such ARMA model.
p.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA models. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Var.Cov.eps=NULL, it checks the suitability of the ARMA models according to the Ljung-Box and the t.test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Var.Cov.eps=NULL, alpha contains the significance level (default is 0.05) which the ARMA models are checked.

Details

If Var.Cov.eps=NULL and the routine is not able to suggest an approximation for Var.Cov.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Var.Cov.eps, the procedure suggested in Domowitz (1982) can be followed.

The implemented procedure particularizes the parametric test in the routine plrm.ancova to the case where is known that the nonparametric components in the corresponding PLR models are null.

Value

A list with a dataframe containing:

Q.beta	value of the test statistic.
p.value	p-value of the corresponding statistic test.

Moreover, if data is a time series and Var.Cov.eps is not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Domowitz, J. (1982) The linear model with stochastic regressors and heteroscedastic dependent errors. Discussion paper No 543, Center for Mathematical studies in Economic and Management Science, Northwestern University, Evanston, Illinois.

Judge, G.G., Griffiths, W.E., Carter Hill, R., Lutkepohl, H. and Lee, T-C. (1980) *The Theory and Practice of Econometrics*. Wiley.

Seber, G.A.F. (1977) *Linear Regression Analysis*. Wiley.

See Also

Other related functions are `np.ancova` and `plrm.ancova`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

data(barnacles2)
data2 <- as.matrix(barnacles2)
data2 <- diff(data2, 12)
data2 <- cbind(data2[,1],1,data2[,-1])

data3 <- array(0, c(nrow(data),ncol(data),2))
data3[,,1] <- data
data3[,,2] <- data2

par.ancova(data=data3)



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data - true null hypothesis

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)

x1 <- matrix(rnorm(200,0,1), nrow=n)
sum1 <- x1%*%beta
epsilon1 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y1 <-  sum1 + epsilon1
data1 <- cbind(y1,x1)

x2 <- matrix(rnorm(200,1,2), nrow=n)
sum2 <- x2%*%beta
epsilon2 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y2 <- sum2 + epsilon2
data2 <- cbind(y2,x2)

data_eq <- array(cbind(data1,data2),c(100,3,2))
```

```
# We apply the test
par.ancova(data_eq, time.series=TRUE)


## Example 2a: dependent data - false null hypothesis
# We generate the data
n <- 100
beta3 <- c(0.05, 0.01)
beta4 <- c(0.05, 0.02)

x3 <- matrix(rnorm(200,0,1), nrow=n)
sum3 <- x3%*%beta3
epsilon3 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y3 <-  sum3 + epsilon3
data3 <- cbind(y3,x3)

x4 <- matrix(rnorm(200,1,2), nrow=n)
sum4 <- x4%*%beta4
epsilon4 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y4 <-  sum4 + epsilon4
data4 <- cbind(y4,x4)

data_neq <- array(cbind(data3,data4),c(100,3,2))

# We apply the test
par.ancova(data_neq, time.series=TRUE)
```

`par.ci` *Confidence intervals estimation in linear regression models*

Description

This routine obtains a confidence interval for the value $a^T * \beta$, by asymptotic distribution and bootstrap, from a sample $(Y_i, X_{i1}, ..., X_{ip}) : i = 1, ..., n$, where:

$$a = (a_1, ..., a_p)^T$$

is an unknown vector,

$$\beta = (\beta_1, ..., \beta_p)^T$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series.

Usage

```
par.ci(data=data, seed=123, CI="AD", B=1000, N=50, a=NULL,
p.arima=NULL, q.arima=NULL, p.max=3, q.max=3, alpha=0.05,
alpha2=0.05, num.lb=10, ic="BIC", Var.Cov.eps=NULL)
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the explanatory variables, $X_1, ..., X_p$.
seed	the considered seed.
CI	method to obtain the confidence interval. It allows us to choose between: "AD" (asymptotic distribution), "B" (bootstrap) or "all" (both). The default is "AD".
B	number of bootstrap replications. The default is 1000.
N	Truncation parameter used in the finite approximation of the MA(infinite) expression of ϵ.
a	Vector which, multiplied by beta, is used for obtaining the confidence interval of this result.
p.arima	the considered p to fit the model ARMA(p,q).
q.arima	the considered q to fit the model ARMA(p,q).
p.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
alpha	1 - alpha is the confidence level of the confidence interval. The default is 0.05.
alpha2	significance level used to check (if needed) the ARMA model fitted to the residuals. The default is 0.05.
num.lb	if Var.Cov.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
Var.Cov.eps	n x n matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.

Value

A list containing:

Bootstrap	a dataframe containing ci_inf and ci_sup, the confidence intervals using bootstrap and p_opt and q_opt (the orders for the ARMA model fitted to the residuals).
AD	a dataframe containing ci_inf and ci_sup, the confidence intervals using the asymptotic distribution and p_opt and q_opt (the orders for the ARMA model fitted to the residuals).
pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Liang, H., Hardle, W., Sommerfeld, V. (2000) Bootstrap approximation in a partially linear regression model. *Journal of Statistical Planning and Inference* **91**, 413-426.

You, J., Zhou, X. (2005) Bootstrap of a semiparametric partially linear model with autoregressive errors. *Statistica Sinica* **15**, 117-133.

See Also

A related function is `plrm.ci`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

## Not run: par.ci(data, a=c(1,0,0), CI="all")
## Not run: par.ci(data, a=c(0,1,0), CI="all")
## Not run: par.ci(data, a=c(0,0,1), CI="all")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(123)
# We generate the data
n <- 100
beta <- c(0.5, 2)

x <- matrix(rnorm(200,0,3), nrow=n)
sum <- x%*%beta
sum <- as.matrix(sum)
eps <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.1, n = n)
eps <- as.matrix(eps)
y <-  sum + eps
data_parci <- cbind(y,x)

# We estimate the confidence interval of a^T * beta in the PLR model
## Not run: par.ci(data, a=c(1,0), CI="all")
## Not run: par.ci(data, a=c(0,1), CI="all")
```

`par.est` *Estimation in linear regression models*

Description

This routine computes the ordinary least squares estimate for β from a sample $(Y_i, X_{i1}, ..., X_{ip})$, $i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series.

Usage

```
par.est(data = data)
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the explanatory variables, $X_1, ..., X_p$.

Details

See Seber (1977) and Judge *et al.* (1980).

Value

A vector containing the corresponding estimate.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Judge, G.G., Griffiths, W.E., Carter Hill, R., Lutkepohl, H. and Lee, T-C. (1980) *The Theory and Practice of Econometrics*. Wiley.

Seber, G.A.F. (1977) *Linear Regression Analysis*. Wiley.

See Also

Other related functions are `plrm.beta` and `plrm.est`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

beta <- par.est(data=data)
beta
residuos <- data[,1] - data[,-1]%*%beta
mean(residuos^2)/var(data[,1])

fitted.values <- data[,-1]%*%beta
plot(data[,1], fitted.values, xlab="y", ylab="y.hat", main="y.hat vs y")
abline(0,1)
```

```
# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
beta <- c(0.05, 0.01)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + epsilon
data_ind <- matrix(c(y,x),nrow=100)

# We estimate the parametric component of the PLR model
par.est(data_ind)


## Example 2b: dependent data

set.seed(1234)
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + epsilon
data_dep <- matrix(c(y,x),nrow=100)

# We estimate the parametric component of the PLR model
par.est(data_dep)
```

`par.gof` *Goodness-of-Fit tests in linear regression models*

Description

This routine tests the equality of the vector of coefficients, β, in a linear regression model and a given parameter vector, β_0, from a sample $(Y_i, X_{i1}, ..., X_{ip}) : i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : \beta = \beta_0$, derives from the asymptotic normality of the ordinary least squares estimator of β, this result giving a χ^2-test.

Usage

```
par.gof(data = data, beta0 = NULL, time.series = FALSE,
Var.Cov.eps = NULL, p.max = 3, q.max = 3, ic = "BIC",
num.lb = 10, alpha = 0.05)
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the explanatory variables, $X_1, ..., X_p$.
beta0	the considered parameter vector in the null hypothesis. If NULL (the default), the zero vector is considered.
time.series	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
Var.Cov.eps	n x n matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted linear regression model and, then, it obtains the var-cov matrix of such ARMA model.
p.max	if Var.Cov.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Var.Cov.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Var.Cov.eps=NULL, alpha contains the significance level which the ARMA model is checked. The default is 0.05.

Details

If Var.Cov.eps=NULL and the routine is not able to suggest an approximation for Var.Cov.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Var.Cov.eps, the procedure suggested in Domowitz (1982) can be followed.

The implemented procedure particularizes the parametric test in the routine plrm.gof to the case where is known that the nonparametric component in the corresponding PLR model is null.

Value

A list with a dataframe containing:

Q.beta	value of the test statistic.
p.value	p-value of the corresponding statistic test.

Moreover, if data is a time series and Var.Cov.eps is not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Domowitz, J. (1982) The linear model with stochastic regressors and heteroscedastic dependent errors. Discussion paper No 543, Center for Mathematical studies in Economic and Management Science, Northwestern University, Evanston, Illinois.

Judge, G.G., Griffiths, W.E., Carter Hill, R., Lutkepohl, H. and Lee, T-C. (1980) *The Theory and Practice of Econometrics*. Wiley.

Seber, G.A.F. (1977) *Linear Regression Analysis*. Wiley.

See Also

Other related functions are `np.gof` and `plrm.gof`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

## Example 1.1: false null hypothesis
par.gof(data)
## Example 1.2: true null hypothesis
par.gof(data, beta0=c(0,0.15,0.4))



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(1234)
# We generate the data
n <- 100
beta <- c(0.05, 0.01)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + epsilon
data <- cbind(y,x)

## Example 2a.1: true null hypothesis
par.gof(data, beta0=c(0.05, 0.01))

## Example 2a.2: false null hypothesis
par.gof(data)
```

`plrm.ancova` *Semiparametric analysis of covariance (based on PLR models)*

Description

From samples $(Y_{ki}, X_{ki1}, ..., X_{kip}, t_i) : i = 1, ..., n,\ k = 1, ..., L$, this routine tests the null hypotheses $H0 : \beta_1 = ... = \beta_L$ and $H0 : m_1 = ... = m_L$, where:

$$\beta_k = (\beta_{k1}, ..., \beta_{kp})$$

is an unknown vector parameter;

$$m_k(.)$$

is a smooth but unknown function and

$$Y_{ki} = X_{ki1} * \beta_{k1} + ... + X_{kip} * \beta_{kp} + m(t_i) + \epsilon_{ki}.$$

Fixed equally spaced design is considered for the "nonparametric" explanatory variable, t, and the random errors, ϵ_{ki}, are allowed to be time series. The test statistic used for testing $H0 : \beta_1 = ... = \beta_L$ derives from the asymptotic normality of an estimator of β_k $(k = 1, ..., L)$ based on both ordinary least squares and kernel smoothing (this result giving a χ^2-test). The test statistic used for testing $H0 : m_1 = ... = m_L$ derives from a Cramer-von-Mises-type functional based on different distances between nonparametric estimators of m_k $(k = 1, ..., L)$.

Usage

```
plrm.ancova(data = data, t = t, b.seq = NULL, h.seq = NULL,
w = NULL, estimator = "NW", kernel = "quadratic",
time.series = FALSE, Var.Cov.eps = NULL, Tau.eps = NULL,
b0 = NULL, h0 = NULL, lag.max = 50, p.max = 3, q.max = 3,
ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

`data`	`data[, 1, k]` contains the values of the response variable, Y_k, for each model k $(k = 1, ..., L)$; `data[, 2:(p+1), k]` contains the values of the "linear" explanatory variables, $X_{k1}, ..., X_{kp}$, for each model k $(k = 1, ..., L)$.
`t`	contains the values of the "nonparametric" explanatory (common) variable, t, for each model k $(k = 1, ..., L)$.
`b.seq`	the statistic test for $H0 : \beta_1 = ... = \beta_L$ is performed using each bandwidth in the vector `b.seq`. If `NULL` (the default) but `h.seq` is not `NULL`, it takes `b.seq=h.seq`. If both `b.seq` and `h.seq` are `NULL`, 10 equidistant values between zero and a quarter of the range of t_i are considered.
`h.seq`	the statistic test for $H0 : m_1 = ... = m_L$ is performed using each pair of bandwidths (`b.seq[j], h.seq[j]`). If `NULL` (the default) but `b.seq` is not `NULL`, it takes `h.seq=b.seq`. If both `b.seq` and `h.seq` are `NULL`, 10 equidistant values between zero and a quarter of the range of t_i are considered for both `b.seq` and `h.seq`.
`w`	support interval of the weigth function in the test statistic for $H0 : m_1 = ... = m_L$. If `NULL` (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

`time.series` it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.

`Var.Cov.eps` `Var.Cov.eps[, , k]` contains the `n x n` matrix of variances-covariances associated to the random errors of the regression model k ($k = 1, ..., L$). If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.

`Tau.eps` `Tau.eps[k]` contains the sum of autocovariances associated to the random errors of the regression model k ($k = 1, ..., L$). If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the sum of the autocovariances of such ARMA model.

`b0` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `b0` contains the pilot bandwidth for the estimator of β_k ($k = 1, ..., L$) used for obtaining the residuals to construct the default for `Var.Cov.eps` and/or `Tau.eps`. If `NULL` (the default) but `h0` is not `NULL`, it takes `b0=h0`. If both `b0` and `h0` are `NULL`, a quarter of the range of t_i is considered.

`h0` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, (`b0, h0`) contains the pair of pilot bandwidths for the estimator of m_k ($k = 1, ..., L$) used for obtaining the residuals to construct the default for `Var.Cov.eps` and/or `Tau.eps`. If `NULL` (the default) but `b0` is not `NULL`, it takes `h0=b0`. If both `b0` and `h0` are `NULL`, a quarter of the range of t_i is considered for both `b0` and `h0`.

`lag.max` if `Tau.eps=NULL`, `lag.max` contains the maximum delay used to construct the default for `Tau.eps`. The default is 50.

`p.max` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=`p.max` and 0<=q<=`q.max`. The default is 3.

`q.max` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=`p.max` and 0<=q<=`q.max`. The default is 3.

`ic` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `ic` contains the information criterion used to suggest the ARMA models. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).

`num.lb` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, it checks the suitability of the selected ARMA models according to the Ljung-Box test and the t-test. It uses up to `num.lb` delays in the Ljung-Box test. The default is 10.

`alpha` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `alpha` contains the significance level which the ARMA models are checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic for testing $H0 : m_1 = ... = m_L$ to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m_k(t_i)$.

If `Var.Cov.eps=NULL` and the routine is not able to suggest an approximation for `Var.Cov.eps`, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct `Var.Cov.eps`, the procedure suggested in Aneiros-Perez and Vieu (2013) can be followed.

If Tau.eps=NULL and the routine is not able to suggest an approximation for Tau.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Tau.eps, the procedures suggested in Aneiros-Perez (2008) can be followed.

Expressions for the implemented statistic tests can be seen in (15) and (16) in Aneiros-Perez (2008).

Value

A list with two dataframes:

parametric.test
: a dataframe containing the bandwidths, the statistics and the p-values when one tests $H0 : \beta_1 = ... = \beta_L$.

nonparametric.test
: a dataframe containing the bandwidths b and h, the statistics, the normalised statistics and the p-values when one tests $H0 : m_1 = ... = m_L$.

Moreover, if data is a time series and Tau.eps or Var.Cov.eps are not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G. (2008) Semi-parametric analysis of covariance under dependence conditions within each group. *Aust. N. Z. J. Stat.* **50**, 97-123.

Aneiros-Perez, G. and Vieu, P. (2013) Testing linearity in semi-parametric functional data analysis. *Comput. Stat.* **28**, 413-434.

See Also

Other related functions are plrm.est, par.ancova and np.ancova.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

data(barnacles2)
data2 <- as.matrix(barnacles2)
data2 <- diff(data2, 12)
data2 <- cbind(data2,1:nrow(data2))

data3 <- array(0, c(nrow(data),ncol(data)-1,2))
data3[,,1] <- data[,-4]
data3[,,2] <- data2[,-4]
```

```
t <- data[,4]

plrm.ancova(data=data3, t=t)



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data - true null hypotheses

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)

m1 <- function(t) {0.25*t*(1-t)}
f <- m1(t)
x1 <- matrix(rnorm(200,0,1), nrow=n)
sum1 <- x1%*%beta
epsilon1 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y1 <-  sum1 + f + epsilon1
data1 <- cbind(y1,x1)

x2 <- matrix(rnorm(200,1,2), nrow=n)
sum2 <- x2%*%beta
epsilon2 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y2 <- sum2 + f + epsilon2
data2 <- cbind(y2,x2)

data_eq <- array(c(data1,data2), c(n,3,2))

# We apply the tests
plrm.ancova(data=data_eq, t=t, time.series=TRUE)


## Example 2b: dependent data - false null hypotheses

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m3 <- function(t) {0.25*t*(1-t)}
m4 <- function(t) {0.25*t*(1-t)*0.75}
beta3 <- c(0.05, 0.01)
beta4 <- c(0.05, 0.02)

x3 <- matrix(rnorm(200,0,1), nrow=n)
sum3 <- x3%*%beta3
f3 <- m3(t)
epsilon3 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y3 <-  sum3 + f3 + epsilon3
data3 <- cbind(y3,x3)

x4 <- matrix(rnorm(200,1,2), nrow=n)
sum4 <- x4%*%beta4
f4 <- m4(t)
epsilon4 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
```

```
y4 <-  sum4 + f4 + epsilon4
data4 <- cbind(y4,x4)

data_neq <- array(c(data3,data4), c(n,3,2))

# We apply the tests
plrm.ancova(data=data_neq, t=t, time.series=TRUE)
```

`plrm.beta` *Semiparametric estimate for the parametric component of the regression function in PLR models*

Description

This routine computes estimates for β from a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

The nonparametric component, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series. Ordinary least squares estimation, combined with kernel smoothing, is used.

Usage

```
plrm.beta(data = data, b.seq = NULL, estimator = "NW", kernel = "quadratic")
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; `data[, p+2]` contains the values of the "nonparametric" explanatory variable, t.
`b.seq`	vector of bandwidths for estimating β. If `NULL` (the default), only one estimate of β is computed, the corresponding bandwidth being selected by means of the cross-validation procedure.
`estimator`	allows us the choice between “NW” (Nadaraya-Watson) or “LLP” (Local Linear Polynomial). The default is “NW”.
`kernel`	allows us the choice between “gaussian”, “quadratic” (Epanechnikov kernel), “triweight” or “uniform” kernel. The default is “quadratic”.

Details

The expression for the estimator of β can be seen in page 52 in Aneiros-Perez *et al.* (2004).

Value

A list containing:

BETA	p x length(b.seq) matrix containing the estimate of β for each bandwidth in h.seq.
G	n x p x length(b.seq) array containing the nonparametric estimate of E(X_{ij} | t_i) ($i = 1, ..., n; j = 1, ..., p$) for each bandwidth in b.seq.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression model under long memory dependence. *Bernoulli* **10**, 49-78.

Hardle, W., Liang, H. and Gao, J. (2000) *Partially Linear Models*. Physica-Verlag.

Speckman, P. (1988) Kernel smoothing in partial linear models. *J. R. Statist. Soc. B* **50**, 413-436.

See Also

Other related functions are: plrm.est, plrm.gcv, plrm.cv.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

b.h <- plrm.gcv(data)$bh.opt
ajuste <- plrm.beta(data=data, b=b.h[1])
ajuste$BETA



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)
```

```
# We estimate the parametric component of the PLR model
# (GCV bandwidth)
a <- plrm.beta(data_ind)

a$BETA


## Example 2b: dependent data

set.seed(1234)
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data_dep <- matrix(c(y,x,t),nrow=100)


# We estimate the parametric component of the PLR model
# (CV bandwidth)
b <- plrm.cv(data_dep, ln.0=2)$bh.opt[2,1]
a <-plrm.beta(data_dep, b=b)

a$BETA
```

---

plrm.ci *Confidence intervals estimation in partial linear regression models*

Description

This routine obtains a confidence interval for the value $a^T * \beta$, by asymptotic distribution and bootstrap, $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, where:

$$a = (a_1, ..., a_p)^T$$

is an unknown vector,

$$\beta = (\beta_1, ..., \beta_p)^T$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

The nonparametric component, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series.

Usage

```
plrm.ci(data=data, seed=123, CI="AD", B=1000, N=50, a=NULL,
        b1=NULL, b2=NULL, estimator="NW",
        kernel="quadratic", p.arima=NULL, q.arima=NULL,
        p.max=3, q.max=3, alpha=0.05, alpha2=0.05, num.lb=10,
        ic="BIC", Var.Cov.eps=NULL)
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; data[, p+2] contains the values of the "nonparametric" explanatory variable, t.
seed	the considered seed.
CI	method to obtain the confidence interval. It allows us to choose between: "AD" (asymptotic distribution), "B" (bootstrap) or "all" (both). The default is "AD".
B	number of bootstrap replications. The default is 1000.
N	Truncation parameter used in the finite approximation of the MA(infinite) expression of ϵ.
a	Vector which, multiplied by beta, is used for obtaining the confidence interval of this result.
b1	the considered bandwidth to estimate the confidence interval by asymptotic distribution. If NULL (the default), it is obtained using cross-validation.
b2	the considered bandwidth to estimate the confidence interval by bootstrap. If NULL (the default), it is obtained using cross-validation.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
p.arima	the considered p to fit the model ARMA(p,q).
q.arima	the considered q to fit the model ARMA(p,q).
p.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
alpha	1 - alpha is the confidence level of the confidence interval. The default is 0.05.
alpha2	significance level used to check (if needed) the ARMA model fitted to the residuals. The default is 0.05.
num.lb	if Var.Cov.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
Var.Cov.eps	n x n matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.

Value

A list containing:

Bootstrap	a dataframe containing ci_inf and ci_sup, the confidence intervals using bootstrap; p_opt and q_opt (the orders for the ARMA model fitted to the residuals) and b1 and b2, the considered bandwidths.
AD	a dataframe containing ci_inf and ci_sup, the confidence intervals using the asymptotic distribution; p_opt and q_opt (the orders for the ARMA model fitted to the residuals) and b1, the considered bandwidth.
pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Liang, H., Hardle, W., Sommerfeld, V. (2000) Bootstrap approximation in a partially linear regression model. *Journal of Statistical Planning and Inference* **91**, 413-426.

You, J., Zhou, X. (2005) Bootstrap of a semiparametric partially linear model with autoregressive errors. *Statistica Sinica* **15**, 117-133.

See Also

A related functions is par.ci.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

b.h <- plrm.gcv(data)$bh.opt
b1 <- b.h[1]

## Not run: plrm.ci(data, b1=b1, b2=b1, a=c(1,0), CI="all")
## Not run: plrm.ci(data, b1=b1, b2=b1, a=c(0,1), CI="all")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(123)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {t+0.5}
f <- m(t)

beta <- c(0.5, 2)
x <- matrix(rnorm(200,0,3), nrow=n)
sum <- x%*%beta
```

```
sum <- as.matrix(sum)
eps <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.1, n = n)
eps <- as.matrix(eps)

y <-  sum + f + eps
data_plrmci <- cbind(y,x,t)

## Not run: plrm.ci(data, a=c(1,0), CI="all")
## Not run: plrm.ci(data, a=c(0,1), CI="all")
```

`plrm.cv` *Cross-validation bandwidth selection in PLR models*

Description

From a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, this routine computes, for each l_n considered, an optimal pair of bandwidths for estimating the regression function of the model

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i,$$

where

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$m(.)$$

is a smooth but unknown function. The random errors, ϵ_i, are allowed to be time series. The optimal pair of bandwidths, (`b.opt, h.opt`), is selected by means of the leave-$(2l_n + 1)$-out cross-validation procedure. The bandwidth `b.opt` is used in the estimate of β, while the pair of bandwidths `(b.opt, h.opt)` is considered in the estimate of m. Kernel smoothing, combined with ordinary least squares estimation, is used.

Usage

```
plrm.cv(data = data, b.equal.h = TRUE, b.seq=NULL, h.seq=NULL,
num.b = NULL, num.h = NULL, w = NULL, num.ln = 1, ln.0 = 0,
step.ln = 2, estimator = "NW", kernel = "quadratic")
```

Arguments

`data`	`data[,1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; `data[, p+2]` contains the values of the "nonparametric" explanatory variable, t.
`b.equal.h`	if TRUE (the default), the same bandwidth is used for estimating both β and m.
`b.seq`	sequence of considered bandwidths, `b`, in the CV function for estimating β. If `NULL` (the default), `num.b` equidistant values between zero and a quarter of the range of t_i are considered.
`h.seq`	sequence of considered bandwidths, `h`, in the pair of bandwidths `(b, h)` used in the CV function for estimating m. If `NULL` (the default), `num.h` equidistant values between zero and a quarter of the range of t_i are considered.

num.b	number of values used to build the sequence of considered bandwidths for estimating β. If b.seq is not NULL, num.b=length(b.seq). Otherwise, if both num.b and num.h are NULL (the default), num.b=50 is considered; if num.b is NULL (the default) but num.h is not NULL, then num.b=num.h is considered; if b.equal.h=TRUE (the default) and both num.b and num.h are not NULL and different, the maximum value of num.b and num.h is considered for both.
num.h	pairs of bandwidths (b, h) are used for estimating m, num.h being the number of values considered for h. If h.seq is not NULL, num.h=length(h.seq). Otherwise, if both num.b and num.h are NULL (the default), num.h=50 is considered; if num.h is NULL (the default) but num.b is not NULL, num.h=num.b is considered; if b.equal.h=TRUE (the default) and both num.b and num.h are not NULL and different, the maximum value of num.b and num.h is considered for both.
w	support interval of the weigth function in the CV function. If NULL (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
num.ln	number of values for l_n: after estimating β, $2l_n + 1$ observations around each point t_i are eliminated to estimate $m(t_i)$ in the CV function. The default is 1.
ln.0	minimum value for l_n. The default is 0.
step.ln	distance between two consecutives values of l_n. The default is 2.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the CV function to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

As noted in the definition of num.ln, the estimate of β in the CV function is obtained from all data while, once β is estimated, $2l_n + 1$ observations around each t_i are eliminated to estimate $m(t_i)$ in the CV function. Actually, the estimate of β to be used in time i in the CV function could be done eliminating such $2l_n + 1$ observations too; that possibility was not implemented because both their computational cost and the known fact that the estimate of β is quite insensitive to the bandwidth selection.

The implemented procedure generalizes that one in expression (8) in Aneiros-Perez and Quintela-del-Rio (2001) by including a weight function (see above) and allowing two smoothing parameters instead of only one (see Aneiros-Perez *et al.*, 2004).

Value

bh.opt	dataframe containing, for each ln considered, the selected value for (b,h).
CV.opt	CV.opt[k] is the minimum value of the CV function when de k-th value of ln is considered.
CV	an array containing the values of the CV function for each pair of bandwidths and ln considered.
b.seq	sequence of considered bandwidths, b, in the CV function for estimating β.
h.seq	sequence of considered bandwidths, h, in the pair of bandwidths (b, h) used in the CV function for estimating m.
w	support interval of the weigth function in the CV function.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Aneiros-Perez, G. and Quintela-del-Rio, A. (2001) Modified cross-validation in semiparametric regression models with dependent errors. *Comm. Statist. Theory Methods* **30**, 289-307.

Chu, C-K and Marron, J.S. (1991) Comparison of two bandwidth selectors with dependent errors. *The Annals of Statistics* **19**, 1906-1918.

See Also

Other related functions are: plrm.beta, plrm.est, plrm.gcv, np.est, np.gcv and np.cv.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

aux <- plrm.cv(data, step.ln=1, num.ln=2)
aux$bh.opt
plot.ts(aux$CV[,-2,])

par(mfrow=c(2,1))
plot(aux$b.seq,aux$CV[,-2,1], xlab="h", ylab="CV", type="l", main="ln=0")
plot(aux$b.seq,aux$CV[,-2,2], xlab="h", ylab="CV", type="l", main="ln=1")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)

# We apply the function
a <-plrm.cv(data_ind)
a$CV.opt
```

```
CV <- a$CV
h <- a$h.seq
plot(h, CV,type="l")


## Example 2b: dependent data and ln.0 > 0

set.seed(1234)
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data_dep <- matrix(c(y,x,t),nrow=100)

# We apply the function
a <-plrm.cv(data_dep, ln.0=2)
a$CV.opt

CV <- a$CV
h <- a$h.seq
plot(h, CV,type="l")
```

`plrm.est` *Semiparametric estimates for the unknown components of the regression function in PLR models*

Description

This routine computes estimates for β and $m(newt_j)$ $(j = 1, ..., J)$ from a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i)$: $i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter,

$$m(.)$$

is a smooth but unknown function and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series. Kernel smoothing, combined with ordinary least squares estimation, is used.

Usage

```
plrm.est(data = data, b = NULL, h = NULL, newt = NULL, estimator = "NW",
kernel = "quadratic")
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; data[, p+2] contains the values of the "nonparametric" explanatory variable, t.
b	bandwidth for estimating the parametric part of the model. If both b and h are NULL (the default), it is selected by means of the cross-validation procedure (fixing b=h); if b is NULL (the default) but h is not NULL, b=h is considered.
h	(b,h) is the pair of bandwidths for estimating the nonparametric part of the model. If both b and h are NULL (the default), it is selected by means of the cross-validation procedure (fixing b=h); if b is NULL (the default) but h is not NULL, b=h is considered; if h is NULL (the default) but b is not NULL, h=b is considered.
newt	values of the "nonparametric" explanatory variable where the estimator of m is evaluated. If NULL (the default), the considered values will be the values of data[,p+2].
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

Expressions for the estimators of β and m can be seen in page 52 in Aneiros-Perez *et al.* (2004).

Value

A list containing:

beta	a vector containing the estimate of β.
m.t	a vector containing the estimator of the non-parametric part, m, evaluated in the design points.
m.newt	a vector containing the estimator of the non-parametric part, m, evaluated in newt.
residuals	a vector containing the residuals: Y - X*beta - m.t.
fitted.values	the values obtained from the expression: X*beta + m.t
b	the considered bandwidth for estimating β.
h	(b,h) is the pair of bandwidths considered for estimating m.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Hardle, W., Liang, H. and Gao, J. (2000) *Partially Linear Models*. Physica-Verlag.

Speckman, P. (1988) Kernel smoothing in partial linear models. *J. R. Statist. Soc. B* **50**, 413-436.

See Also

Other related functions are: `plrm.beta`, `plrm.gcv`, `plrm.cv`, `np.est`, `np.gcv` and `np.cv`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

b.h <- plrm.gcv(data)$bh.opt
ajuste <- plrm.est(data=data, b=b.h[1], h=b.h[2])
ajuste$beta
plot(data[,4], ajuste$m, type="l", xlab="t", ylab="m(t)")

plot(data[,1], ajuste$fitted.values, xlab="y", ylab="y.hat", main="y.hat vs y")
abline(0,1)

mean(ajuste$residuals^2)/var(data[,1])



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)

# We estimate the components of the PLR model
# (CV bandwidth)
a <- plrm.est(data_ind)

a$beta

est <- a$m.t
plot(t, est, type="l", lty=2, ylab="")
```

```
points(t, 0.25*t*(1-t), type="l")
legend(x="topleft", legend = c("m", "m hat"), col=c("black", "black"), lty=c(1,2))


## Example 2b: dependent data
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data_dep <- matrix(c(y,x,t),nrow=100)

# We estimate the components of the PLR model
# (CV bandwidth)
h <- plrm.cv(data_dep, ln.0=2)$bh.opt[3,1]
a <- plrm.est(data_dep, h=h)

a$beta

est <- a$m.t
plot(t, est, type="l", lty=2, ylab="")
points(t, 0.25*t*(1-t), type="l")
legend(x="topleft", legend = c("m", "m hat"), col=c("black", "black"), lty=c(1,2))
```

`plrm.gcv` *Generalized cross-validation bandwidth selection in PLR models*

Description

From a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, this routine computes an optimal pair of bandwidths for estimating the regression function of the model

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i,$$

where

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$m(.)$$

is a smooth but unknown function. The optimal pair of bandwidths, `(b.opt, h.opt)`, is selected by means of the generalized cross-validation procedure. The bandwidth `b.opt` is used in the estimate of β, while the pair of bandwidths `(b.opt, h.opt)` is considered in the estimate of m. Kernel smoothing, combined with ordinary least squares estimation, is used.

Usage

```
plrm.gcv(data = data, b.equal.h = TRUE, b.seq=NULL, h.seq=NULL,
num.b = NULL, num.h = NULL, estimator = "NW", kernel = "quadratic")
```

Arguments

data	data[,1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; data[, p+2] contains the values of the "nonparametric" explanatory variable, t.
b.equal.h	if TRUE (the default), the same bandwidth is used for estimating both β and m.
b.seq	sequence of considered bandwidths, b, in the GCV function for estimating β.
h.seq	sequence of considered bandwidths, h, in the pair of bandwidths (b, h) used in the GCV function for estimating m.
num.b	number of values used to build the sequence of considered bandwidths for estimating β. If b.seq is not NULL, num.b=length(b.seq). Otherwise, if both num.b and num.h are NULL (the default), num.b=50 is considered; if num.b is NULL (the default) but num.h is not NULL, then num.b=num.h is considered; if b.equal.h=TRUE (the default) and both num.b and num.h are not NULL and different, the maximum value of num.b and num.h is considered for both.
num.h	pairs of bandwidths (b, h) are used for estimating m, num.h being the number of values considered for h. If h.seq is not NULL, num.h=length(h.seq). Otherwise, if both num.b and num.h are NULL (the default), num.h=50 is considered; if num.h is NULL (the default) but num.b is not NULL, num.h=num.b is considered; if b.equal.h=TRUE (the default) and both num.b and num.h are not NULL and different, the maximum value of num.b and num.h is considered for both.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

The implemented procedure generalizes that one in page 423 in Speckman (1988) by allowing two smoothing parameters instead of only one (see Aneiros-Perez *et al.*, 2004).

Value

bh.opt	selected value for (b,h).
GCV.opt	minimum value of the GCV function.
GCV	matrix containing the values of the GCV function for each pair of bandwidths considered.
b.seq	sequence of considered bandwidths, b, in the GCV function for estimating β. If b.seq was not input by the user, it is composed by num.b equidistant values between zero and a quarter of the range of t_i.
h.seq	sequence of considered bandwidths, h, in the pair of bandwidths (b, h) used in the GCV function for estimating m. If h.seq was not input by the user, it is composed by num.h equidistant values between zero and a quarter of the range of t_i.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Green, P. (1985) Linear models for field trials, smoothing and cross-validation. *Biometrika* **72**, 527-537.

Speckman, P. (1988) Kernel smoothing in partial linear models *J. R. Statist. Soc. B* **50**, 413-436.

See Also

Other related functions are: `plrm.beta`, `plrm.est`, `plrm.cv`, `np.est`, `np.gcv` and `np.cv`.

Examples

```
# EXAMPLE 1: REAL DATA

data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

aux <- plrm.gcv(data)
aux$bh.opt
plot(aux$b.seq, aux$GCV, xlab="h", ylab="GCV", type="l")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)

# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)

# We obtain the optimal bandwidths
a <-plrm.gcv(data_ind)
a$GCV.opt

GCV <- a$GCV
h <- a$h.seq
plot(h, GCV,type="l")
```

plrm.gof *Goodness-of-Fit tests in PLR models*

Description

From a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, this routine tests the null hypotheses $H_0 : \beta = \beta_0$ and $H_0 : m = m_0$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter,

$$m(.)$$

is a smooth but unknown function and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

Fixed equally spaced design is considered for the "nonparametric" explanatory variable, t, and the random errors, ϵ_i, are allowed to be time series. The test statistic used for testing $H0 : \beta = \beta_0$ derives from the asymptotic normality of an estimator of β based on both ordinary least squares and kernel smoothing (this result giving a χ^2-test). The test statistic used for testing $H0 : m = m_0$ derives from a Cramer-von-Mises-type functional distance between a nonparametric estimator of m and m_0.

Usage

```
plrm.gof(data = data, beta0 = NULL, m0 = NULL, b.seq = NULL,
h.seq = NULL, w = NULL, estimator = "NW", kernel = "quadratic",
time.series = FALSE, Var.Cov.eps = NULL, Tau.eps = NULL,
b0 = NULL, h0 = NULL, lag.max = 50, p.max = 3, q.max = 3,
ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

data	data[,1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the "linear" explanatory variables $X_1, ...X_p$; data[, p+2] contains the values of the "nonparametric" explanatory variable, t.
beta0	the considered parameter vector in the parametric null hypothesis. If NULL (the default), the zero vector is considered.
m0	the considered function in the nonparametric null hypothesis. If NULL (the default), the zero function is considered.
b.seq	the statistic test for $H0 : \beta = \beta_0$ is performed using each bandwidth in the vector b.seq. If NULL (the default) but h.seq is not NULL, it takes b.seq=h.seq. If both b.seq and h.seq are NULL, 10 equidistant values between zero and a quarter of the range of t_i are considered.
h.seq	the statistic test for $H0 : m = m_0$ is performed using each pair of bandwidths (b.seq[j], h.seq[j]). If NULL (the default) but b.seq is not NULL, it takes h.seq=b.seq. If both b.seq and h.seq are NULL, 10 equidistant values between zero and a quarter of the range of t_i are considered for both b.seq and h.seq.

w	support interval of the weigth function in the test statistic for $H0 : m = m_0$. If NULL (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
time.series	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
Var.Cov.eps	n x n matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.
Tau.eps	it contains the sum of autocovariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the sum of the autocovariances of such ARMA model.
b0	if Var.Cov.eps=NULL and/or Tau.eps=NULL, b0 contains the pilot bandwidth for the estimator of β used for obtaining the residuals to construct the default for Var.Cov.eps and/or Tau.eps. If NULL (the default) but h0 is not NULL, it takes b0=h0. If both b0 and h0 are NULL, a quarter of the range of t_i is considered.
h0	if Var.Cov.eps=NULL and/or Tau.eps=NULL, (b0, h0) contains the pair of pilot bandwidths for the estimator of m used for obtaining the residuals to construct the default for Var.Cov.eps and/or Tau.eps. If NULL (the default) but b0 is not NULL, it takes h0=b0. If both b0 and h0 are NULL, a quarter of the range of t_i is considered for both b0 and h0.
lag.max	if Tau.eps=NULL, lag.max contains the maximum delay used to construct the default for Tau.eps. The default is 50.
p.max	if Var.Cov.eps=NULL and/or Tau.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL and/or Tau.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Var.Cov.eps=NULL and/or Tau.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Var.Cov.eps=NULL and/or Tau.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Var.Cov.eps=NULL and/or Tau.eps=NULL, alpha contains the significance level which the ARMA model is checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic for testing $H0 : m = m_0$ to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

If `Var.Cov.eps=NULL` and the routine is not able to suggest an approximation for `Var.Cov.eps`, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct `Var.Cov.eps`, the procedure suggested in Aneiros-Perez and Vieu (2013) can be followed.

If `Tau.eps=NULL` and the routine is not able to suggest an approximation for `Tau.eps`, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct `Tau.eps`, the procedures suggested in Aneiros-Perez (2008) can be followed.

The implemented procedures generalize those ones in expressions (9) and (10) in Gonzalez-Manteiga and Aneiros-Perez (2003) by allowing some dependence condition in $(X_{i1}, ..., X_{ip}) : i = 1, ..., n$ and including a weight function (see above), respectively.

Value

A list with two dataframes:

`parametric.test`
: a dataframe containing the bandwidths, the statistics and the p-values when one tests $H0 : \beta = \beta_0$

`nonparametric.test`
: a dataframe containing the bandwidths b and h, the statistics, the normalised statistics and the p-values when one tests $H0 : m = m_0$

Moreover, if `data` is a time series and `Tau.eps` or `Var.Cov.eps` are not especified:

`pv.Box.test`	p-values of the Ljung-Box test for the model fitted to the residuals.
`pv.t.test`	p-values of the t.test for the model fitted to the residuals.
`ar.ma`	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G. (2008) Semi-parametric analysis of covariance under dependence conditions within each group. *Aust. N. Z. J. Stat.* **50**, 97-123.

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Aneiros-Perez, G. and Vieu, P. (2013) Testing linearity in semi-parametric functional data analysis. *Comput. Stat.* **28**, 413-434.

Gao, J. (1997) Adaptive parametric test in a semiparametric regression model. *Comm. Statist. Theory Methods* **26**, 787-800.

Gonzalez-Manteiga, W. and Aneiros-Perez, G. (2003) Testing in partial linear regression models with dependent errors. *J. Nonparametr. Statist.* **15**, 93-111.

See Also

Other related functions are `plrm.est`, `par.gof` and `np.gof`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

plrm.gof(data)
plrm.gof(data, beta0=c(-0.1, 0.35))



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
f.function <- function(u) {0.25*u*(1-u)}

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data <- cbind(y,x,t)

## Example 2a.1: true null hypotheses
plrm.gof(data, beta0=c(0.05, 0.01), m0=f.function, time.series=TRUE)

## Example 2a.2: false null hypotheses
plrm.gof(data, time.series=TRUE)
```

Index